Dynamic reconfiguration and low power design : towards self-adaptive massively parallel embedded systems

Rabie Ben Atitallah

Université de Valenciennes et du Hainaut-Cambrésis

THÈSE

Soutenance prévue publiquement le 04 December 2014

pour obtenir le titre de

L'HABILITATION À DIRIGER DES RECHERCHES EN INFORMATIQUE

par

Rabie BEN ATITALLAH

Dynamic reconfiguration and low power design : towards self-adaptive massively parallel embedded systems

Composition du jury

Rapporteurs :	Patrick GARDA	Professeur	Université de Pierre et Marie Curie
	Dragomir MILOJEVIC	Professeur	École Polytechnique de Bruxelles
	Loïc LAGADEC	Professeur	ENSTA Bretagne
Examinateurs :	Michel AUGUIN	Directeur de Recherche	Université Nice Sophia Antipolis
	Jean-Luc DEKEYSER	Professeur	Université de Lille1
	Abdelhakim ARTIBA	Professeur	Université de Valenciennes

UNIVERSITÉ DE VALENCIENNES ET DU HAINAUT-CAMBRÉSIS
Laboratoire d'Automatique, de Mécanique et d'Informatique industrielles et Humaines 8201
UVHC, Le Mont Houy, 59313 Valenciennes Cedex 9
Tél. : +33 (0)3 27 51 13 50 – Télécopie : +33 (0)3 27 51 19 40

Acknowledgements

First of all, I am very grateful to Prof. Patrick GARDA from the University of Pierre et Marie Curie (France), Prof. Dragomir MILOJEVIC from École polytechnique de Bruxelles (Belgium) and Prof. Loïc Lagadec from ENSTA Bretagne for having kindly accepted to serve as reviewers on my Habilitation defense committee.

The summary

This thesis summarizes my experience in the attractive academic research of embedded systems that started at November 2004 as a PhD student at the University of Lille1. Actually, I prepared my PhD thesis in Computer Science in the frame of the European Interreg project ModEasy that stands for MOdel Driven dEsign for Automotive Safety embedded systems. In March 2008, I defended my research work entitled "Multiprocessor system-on-chip modeling and simulation - performance and energy consumption estimation". This experience gave me the opportunity to discover the embedded system domain that enables innovation through intelligent products covering a large spectrum of industries (multimedia, communication, transportation, etc.). During this period, I made profit of being a member of the DaRT [1] INRIA team to learn about different research areas. Indeed, this team gathered several researchers from different backgrounds (parallel architecture, modeling, compilation, etc.), all contributing in one project called *Gaspard*. At the end of my PhD, I succeeded to efficiently contribute in this project by publishing with all the team members and by presenting the *Gaspard* tool at FDL [2] 2008 in a tutorial session. This successful experience was my main motivation to continue in the academic research domain adopting the topic of *Dynamic reconfiguration and low power design : towards self-adaptive massively parallel embedded systems*. In the following, I will detail the compelling story of my research career path.

The birth era. In November 2004 when I joined the INRIA DaRT team-project led by Professor Jean-Luc Dekeyser in Lille, the team was only two years old. Its main research topic was about the co-design of System-on-Chip (SoC) for intensive signal processing applications. The main challenges addressed by this project were the following i) The definition of a UML profile for SoC co-modeling, ii) The compilation devoted to data-parallel structures for efficient mapping on multiprocessor platforms, and iii) The functional simulation using SystemC. Based on my fundamentals in electronic domain, I linked between the research topics of DaRT and the low level design. With the help of my supervisors, I focused on Multiprocessor System-on-Chip (MPSoC) simulation and Design Space Exploration (DSE).

In the first design steps, MPSoC simulation has an important impact in reducing the time to market of the final product. However, MPSoC have become more and more complex and heterogeneous. Consequently, traditional approaches for system simulation at lower levels cannot adequately support the complexity of future MPSoC. In my PhD thesis, I proposed a framework composed of several simulation levels. This enables early performance evaluation in the design flow. The proposed framework is useful for DSE and permits to find rapidly the most adequate Architecture/Application configuration. In the first part of the thesis, I presented an efficient simulation tool composed of three levels that offer several performance/energy trade-offs. The three levels are differentiated by the accuracy of architectural descriptions based on the SystemC-TLM standard. In the second part, I was interested in the MPSoC energy consumption. For this, I enhanced the simulation framework with flexible and accurate energy consumption models. Finally in the third part, a compilation chain based on a Model Driven Engineering (MDE) approach was developed and integrated in the Gaspard environment. This chain allows automatic SystemC code generation from

1. DaRT team-project at INRIA Lille Nord Europe (2004-2012)
2. FDL : Forum on specification & Design Languages, 2008, Stuttgart, Germany

high level MPSoC modeling.

Since 2005, I started teaching micro-architecture and assembly language for undergraduate students at University of Valenciennes and Hainaut-Cambrésis (UVHC).

The emergence era. After my PhD defence, I got my first Post-doc position at INRIA Lille-Nord Europe in the frame of the FUI ("Fonds Unique Interministériel" in french) Ter@ops project under the supervision of Professor Pierre Boulet. This project brought together the main french industrial actors in embedded systems (Thales, Thomson, EADS, MBDA, Dassault, RENAULT, VALEO, CEA, etc.) in addition to the academic laboratories (INRIA, ENSTA, IEF, etc.). The objective of the project was to define a multi-domain massively parallel architecture and its corresponding tools. The proposed architecture was able to embed heterogeneous accelerators optimised for the target application domains. A design environment was needed to optimise and to map the application on the parallel architecture. Due to the experience I gained during my PhD in the field of embedded multiprocessor system design, I was invited to participate in this project. The subject of my Post-doc dealt with automatic code generation of an execution model described with SystemC from the standard MARTE profile (Model and Analysis Real-Time Embedded System). Indeed, in the Ter@ops project a particular attention was given to the use of standards to facilitate the interoperability between the different tools of the partners. During this work, the MARTE profile was extended to be able to specify the deployment of software and hardware components. Moreover, the toolchain developed during my thesis was modified to generate a SystemC simulation for multi-processor from the MARTE profile, which led to a new version of our *Gaspard* tool. Despite the lack of scientific mobility, this Post-doc allowed me to benefit from several advantages. First, highlighting my PhD work in the context of an industrial project. Second, making contacts with the industrial actors during the regular meetings of the Ter@ops project. Hence, I succeeded to identify better the challenges and to expand my knowledge in the embedded system domain.

Since October 2008, I got a second Post-doc fellowship for one year at LAMIH [3] laboratory funded by the PrimaCare ANR [4] project. PrimaCare aims at designing Driver Assistance System (DAS) for automotive based on Multiple Target Tracking (MTT) algorithms. My research subject focused on the optimisation and the improvement of the MTT system as an application specific System-on-Chip (SoC) implemented on FPGA. I developed a heterogeneous architecture (Microblaze with hardware accelerators) taking into account the constraints of the automotive embedded system in terms of performance, low cost and reliability.

During this period in addition to my research activities, I supervised several master research students and taught embedded system design at UVHC and Lille1 at the master level. Scientific dissemination was also a part of my activities by participating in the thematic days organised by the national GdRs ("Groupe de Recherche" in french) SoC-SiP (System-on-Chip System-in-Package) and ASR ("Architecture, Systèmes et Réseaux" in french). As a member of the HiPEAC [5] European Network of Excellence, I participated in the main organised events such as summer schools and conferences. In March 2009, I contributed in the writing of the OpenPeople proposal for a request of ANR funding.

3. LAMIH : Laboratoire d'Automatique, de Mécanique et d'Informatique industrielles et Humaines UMR CNRS/UVHC 8201

4. ANR : Agence Nationale de Recherche

5. http ://www.hipeac.net/

OpenPeople stands for Open Power and Energy Optimization PLatform and Estimator. In parallel, I started looking for a permanent position as an Associate Professor in France and I succeeded to join the scientific staff of UVHC.

The maturity era. Since September 2009, I obtained a full-time position as an Associate Professor at UVHC and a member of LAMIH within the DIM [6] team that gathers researchers from different backgrounds (operational research, artificial intelligence, embedded systems, etc.). For five years, I have been investigating, with my colleagues, this complementarity in order to provide a satisfactory answer to the crucial design challenges of modern embedded systems targeting intelligent transportation applications. I am particularly grateful to Professor Abdelhakim Artiba (optimisation and scheduling) and Associate Professor David Duvivier (decision making and simulation) for their fruitful collaborations on this topic (2 PhD students co-supervising, research projects, publications, etc.).

With an agreement between the UVHC and INRIA, I had the status of Associated Researcher at INRIA Lille Nord Europe within the DaRT team until the end of 2012. I managed the contributions of the team in the OpenPeople ANR project and I co-supervised the PhD thesis of Santhosh Kumar Rethinagiri [100] entitled "System-level power estimation methodology for MPSoC based platforms". In 2012, I also participated in the creation of the new INRIA project-team DreamPal (Dynamic Reconfigurable Massively Parallel Architectures and Languages). The DreamPal team addresses the following topics : designing massively parallel dynamically reconfigurable architectures, proposing execution models as well as dedicated programming languages for them, and designing software engineering tools for those languages. I am particularly grateful to Professor Jean-Luc Dekeyser for our successful collaboration on this topic (4 PhD students co-supervising, research projects, patents, publications, etc.).

Today, some of my research works were completed (Santhosh Kumar Rethinagiri's PhD, George Afonso's PhD, Omar Souissi's PhD) ; others are in progress (Venkatasubramanian Viswanathan's PhD, Karim Mohamed Ali's PhD, and Wissem Chouchene's PhD) or still at an exploratory stage. The obtained results are currently relevant enough to contribute to the maturity of my research activity. I am deeply indebted to all my PhD and Master students, and Post-doc colleagues, who contributed actively to the different results summarized in this document.

The content of this document mainly relies on the results of three PhDs : Santhosh Kumar Rethinagiri (December 2009 - March 2013), George Afonso [49] (August 2010 - July 2013), and Venkatasubramanian Viswanathan (February 2012 - January 2015). While for the PhD of Omar Souissi (October 2011 - September 2014) is targeting scheduling techniques for path planning on a high performance heterogeneous CPU/FPGA architecture. But for the content coherence, the results of this thesis are not included in this document.

During this period, significant highlights can be cited : co-supervising of three graduated PhD students, patents, publishing journal papers in IEEE transactions and ACM transactions, acquiring research contracts with ANR (Agence Nationale de Recherche) and industrial partners (Airbus Group, Airbus Helicopters, Nolam Embedded Systems, etc.), winning best paper award, being a member of the steering committee of GdR ASR, participating in the organization of scientific conferences, being responsible for technology transfer action with Airbus Helicopters, etc.

6. DIM (Decision, Interaction, Mobility) : http ://www.univ-valenciennes.fr/LAMIH/en/dim-overview

Rationale of this synthesis document. This document is a summary of my scientific contributions since my Post-doc. It aims to show coherently how my research activities on the design of dynamic reconfigurable system and low power design are leading towards self-adaptive massively parallel embedded systems.

Modern embedded applications are becoming more and more sophisticated and resource demanding. The concerned applications covers several domains such as avionic, multimedia, etc. The computation requirements for such systems are very important in order to meet real-time constraints and high quality of services. Furthermore, power consumption becomes a critical pre-design metric in complex embedded systems. Runtime adaptivity according to the system requirements or the environment variations is also a challenging characteristic which brings additional complexity to the design flow.

For addressing the above challenges, my research works are directed towards designing parallel and dynamic architecture to deal with the potential parallelism and the adaptivity inherent from the application. Mainly, we rely on multiprocessor and reconfigurable systems that could offer better power efficiency. According to the application context, the architecture can be homogeneous or heterogeneous to satisfy a specific need or to optimise better some parameters. In each step of our work, we defined the efficient dynamic execution model that handles the heterogeneity and the parallelism concepts. To cope with the development complexity, we proposed the efficient design methodology and we provided the appropriate framework regarding the application domain. This complexity is reduced by the means of Electronic System Level (ESL) tools such as simulator, power estimator, and prototyping environment.

How to consider the contents of this document. All the results reported in Chapters 2, 3 and 4 have been already published or under revision in peer-reviewed journals and conferences. In order to easily assess my achievements, a section named *discussion* is provided at the end of each chapter. Therefore, a major part of the presented material comes from the related publications. Most of my papers cited as references in this document are online available on the websites of editors or on my website.

Table of contents

Chapter 1

Introduction

1.1 Context of the work

Sophisticated embedded systems are becoming wide spread today in aerospace, automotive, avionic, and defence industries. They are responsible for control, collision avoidance, driver assistance, target tracking, navigation and communications, amongst other functions. According to the characteristics of these functionalities, high computation rates should be well-delivered while carrying-out intensive signal processing. Furthermore, these embedded systems often operate in uncertain environments. So, they should adapt their functioning mode according to the environmental conditions to provide reliability, fault tolerance, deterministic timing guarantees, and energy efficiency. Undoubtedly, the essential feature of systems to reconfigure themselves (at the hardware or the software level) at run-time comes with additional complexity in the different design flow steps.

The design of these sophisticated embedded systems calls for several teams with different domain experts covering electronics, architecture, software engineering, etc. ***We are in a new era promoting for the dynamicity of heterogeneous and parallel processing***. Academic and industrial researchers must address the challenges inherent from this new trend at all the design steps. The scope of my research topics covers mainly the architecture, the execution model, and the design methodology. This document brings some answer elements to address such challenges. It presents a summary of my contributions since March 2008 on dynamic reconfiguration and low power design : towards self-adaptive massively parallel embedded systems. These works was achieved successively at the LAMIH laboratory and INRIA Lille-Nord Europe in collaborations with colleagues from different universities and countries. This introductory chapter is organized as follows : Section 1.2 presents the major trends and challenges in embedded system design according to my research topics. Section 1.3 summarizes my scientific contributions and Section 1.4 describes the outline of the document.

1.2 Trends and challenges

1.2.1 Parallelism and heterogeneity for energy efficiency

In the first decade of the 21st century, computing systems have shifted from a PC-centric approach to a highly integrated System-on-Chip (SoC) making profit from the gigantic number of transistors offered by deep submicron technologies. SoC may contain processors, hierarchical memories, dedicated accelerators, peripherals, etc. As example of recent realization (2013), we quote the Microsoft Xbox One SoC that integrates 5 Billion transistors in 363 mm^2 die using a process of 28 nm which offers a performance of 1.31 TFlops. A typical usage of SoC is in the area of embedded systems. In the last decade, a second observation had also marked computing systems ; ***the end of the dominance of the single microproessor architecture***. Indeed, the growth in the performance for a single-processor stopped because the increase in the clock speed hit the major constraint of ***power consumption limits***. This issue is inherent from the Complementary Metal-Oxide Semiconductor (CMOS) technology largely used in Integrated Circuit (IC) manufacturing [52]. By 2004, the long-fruitful strategy of scaling down the size of CMOS circuits, reducing the supply voltage, and increasing the clock rate had become infeasible [87]. Recently, the ITRS [97] and HiPEAC [83] roadmaps promote that ***power defines performance*** and ***power is the wall***.

To overcome this obstacle, a new era, in which ***parallelism*** dominates the cutting-edge of embedded architecture, appeared [87]. As a result, the whole computing domain is being forced to switch from a focus on performance-centric sequential computation to energy-efficient parallel computation. This switch is driven by the energy efficiency of using many slower parallel processors instead of a single high-speed one [83]. This has led to the design of ***Multiprocessor System-on-Chip (MPSoC)*** that integrates multiple cores or processors on a single die [143]. As an example of commercial platforms based on such architecture, we quote the Texas Instrument OMAP4 platform [7] embedding a dual core ARM Cortex A9 MP-core processor and the NVIDIA Tegra [8] processor integrating a quad-core ARM Cortex A15. Kalray Incorporation [9] proposes the Multi-Purpose Processor Array (MPPA) that integrates 256 processors onto a single silicon chip through a high bandwidth Network on Chip. Unfortunately, these trends are adequate only for a given range of applications particularly in systematic signal processing domain due to the general purpose processor used in theses architectures. This was not enough for other applications where more performance and higher energy-efficiency are required. This has led to the combination of many specialized computing nodes to increase efficiency. Hence, hardware designers switch to the ***heterogeneous era***. For instance, the the Microsoft Xbox One SoC features two AMD quad-core Jaguar x86 modules and 14 AMD GCN CUs (Graphics Core Next Compute Units) with 64 stream processors per unit. P2012 [64] is another example of SoC based on multiple globally asynchronous, locally synchronous (GALS) clusters featuring up to 16 processors. Each processor can be customized at the design time with modular extensions (vector units, floating point units, special purpose instructions). Clusters can easily become heterogeneous computing engines thanks to the integration of coarse grained hardware accelerators.

In parallel, FPGA reconfigurable circuits have emerged as a privileged target platform to implement intensive signal processing applications. Indeed, FPGAs have the benefits of high speed and adaptability to the application constraints with a reduced performance per watt comparing to the General Purpose Processors (GPP). They offer an inexpensive and fast programmable hardware on some of the most advanced fabrication processes. Furthermore, FPGA technology enables today to implement massively parallel architectures due to the huge number of programmable logic fabrics available on the chip. Hardware designers are directed more and more towards heterogeneous architectures that gather a variable number of processors (hardcore or softcore) coupled with specialized hardware accelerators in order to address specific application constraints (timing constraints, power consumption, etc.). Such architectures can be customized at runtime using the Dynamic Partial Reconfiguration (DPR) feature which favours the reconfigurable technology to be a potential solution in order to implement adaptive embedded systems. The Xilinx Zynq 7000 Extensible Processing Platform (EPP) is an example of such circuit embedding a dual core ARM Cortex A9 processor and tens of thousands of programmable gate arrays. The Zynq 7000 combines the software programmability of a processor with the hardware programmability of an FPGA, resulting in unrivaled levels of system performance, flexibility, scalability while providing system benefits in terms of power reduction, lower cost with fast time to market.

With the management of the parallelism and dynamicity intrinsic in the application, the system designer will have several implementation choices such as sequential software,

7. http ://www.ti.com/
8. http ://www.nvidia.fr/object/tegra-k1-processor-fr.html
9. http ://www.kalray.eu/

parallel software, hardware/software, parallel hardware, and dynamic hardware. The adequate choice will depend mainly on the application requirements in terms of performance and energy consumption. ***We must invest in research and development of parallel and heterogeneous power-efficient systems driven by applications. Dynamicity at the hardware or the software level is a characteristic of these systems***. This research must include the definition of an appropriate execution model, an efficient energy-aware design methodology, and their related tools.

1.2.2 Design methodology and development tools

The trends towards ***parallel and heterogeneous power-efficient systems*** requires the definition of new execution and programming models, revisiting the design methodology and the development of CAD tools. Dynamicity brings an additional complexity in all the design layers.

For the programming model, the existing applications are not written in a way to take the advantage of parallel and heterogeneous architectures. Software developers must re-design large parts of their applications at astronomical cost. This was the case for the P2012 [64] SoC while programming heterogeneous clusters which makes the software development of such systems very difficult. Designers must use standard OpenCL and OpenMP parallel codes as well as proprietary Native Programming Model (NPM) for the software part and Hardware Description Language (HDL) to code hardware accelerators. ***There is a need to converge towards a unified programming model to express the parallelism, the heterogeneity, and the dynamicity at the same level***.

For the execution model, challenges start with the definition of the appropriate computing nodes (Hardcore processor, Softcore processor, hardware accelerator, etc.) for a given application using a target technology (ASIC, FPGA, etc.). ***Answers about generalised vs. dedicated, sequential vs. parallel, and static vs. dynamic should be given***. The runtime system reconfiguration comes also with additional questions : ***software or hardware reconfiguration ? who will reconfigure (the operating system or a hardware controller), when to configure ?*** With the new sub-micron technologies, the huge number of transistors leads to a consumption that can not be offered by the power supply of the circuit, which imposes only a partial use of the total capacity. Such an idea refers to ***the dark silicon*** [86] where the system should be configured to exploit the strict necessary chip resources and to stay within the power limit according to the application requirements. With the DPR feature offered by FPGA technology, architectures can be customized at runtime, a reconfiguration that can be done for a subset of the system to offer a better Quality of Service (QoS), to use efficiently the available resources or to minimize the power. So, ***resources can be activated at the demand of the application to satisfy functional and non-functional requirements***. This new paradigm opens many opportunities for research to investigate execution models for parallel and dynamic reconfigurable architectures and the dedicated tools used for mapping intensive signal processing applications on these architectures.

For the design methodology, embedded systems often requires ***hardware/software co-design*** for products that will be commercialised only with a volume of thousands or tens of thousands of units. This small volume makes design too expensive with existing tools. To overcome this challenge, more efficient and rapid design tools are needed that can reduce the time to adapt or create new designs, particularly through Electronic System Level (ESL) approaches that favour the reuse of Intellectual Property (IP) blocks and lead to more adap-

tive systems. Today, minimization of power consumption drives IC design ; for this reason, we need the development of tools for power estimation and optimisation at the system level. Unfortunately, we are in the face of extremely challenging requirements such as a seamless power-aware design methodology that relies on accurate and fast system power modeling and integrates efficient power optimisation techniques. An early and reliable Design Space Exploration (DSE) strategy should be defined to reduce the design complexity.

1.2.3 Industrial manufacturing process

In large industries (automotive, avionic, etc.), real-time embedded computing systems are increasingly used. In the face of power wall, performance requirements and demands for higher flexibility, hardware designers are directed towards reconfigurable and heterogeneous computing that offers high computation rates per watt and adaptability to the application constraints. We highlight that the design process of embedded systems in such industries is totally different comparing to mobile or multimedia domains. Considering reconfigurable computing for example in the avionic design process leads to several challenges for system developers. Indeed, such technology should be validated along the development cycle starting with simulation tools, passing through the test benches and finishing with the integration phase. These design steps should rely on common frameworks in the future supported by cutting-edge hardware architectures. For each step, reconfigurable technology can play an essential role to achieve better performances, more adaptive systems and cost-effective solutions. In this perspective, it is necessary to consolidate the usual design cycle with a solid system approach allowing to early check/validate the adequacy and the consistency of the reconfigurable technology to be embedded in large industries.

The technical areas of simulation, test and integration systems are currently in an unavoidable convergence path. In fact, for high performance or embedded computing, the academic and industrial researchers share the same vision about the trends towards ***parallel and heterogeneous systems***. We need also ***dynamicity*** in the all design process in order to satisfy specific functionalities of the system or to converge towards unified development tools. Energy has become the primary limiting factor in the development of all systems, whether due to the cost of energy and cooling in large systems or due to battery life and reliability factor in embedded devices. We join again the same challenge of Section 1.2.1, however with different design constraints due to the application context covering high performance rather than embedded computing. This research must include also the definition of an appropriate execution model and the appropriate development tools.

1.3 Contributions

From the trends presented in Section 1.2, the nature of future embedded systems are directed more and more towards parallel and heterogeneous architecture where dynamicity will be a key feature. Within this context, my scientific contributions are following two main directions as shown in Figure 1.1 : the low power design and execution models for dynamically reconfigurable systems. These two directions will lead to the definition of our future project about self-adaptive massively parallel embedded systems.

For the first direction, early we advocated for the usage of the reconfigurable technology in the industrial manufacturing process in order to validate such technology along

the development cycle. We proposed an appropriate execution model supported by heterogeneous CPU/FPGA system where dynamicity is a key feature in order to achieve better performances, more adaptive systems and cost-effective solutions. During time, our vision evolves influenced by technological ascendancy and the requirements of nowadays sophisticated applications. We defined a new massively parallel dynamically reconfigurable architecture execution model. For the development assessment, we conceived a Multi-FPGA based platform that supports parallel reconfiguration. Finally, we introduced the original HoMade processor with its programming model that handles the parallelism and the dynamicity at the same level.

For the second direction, we focused on the design of power-efficient MPSoC devices by means of the development of an efficient design methodology and the corresponding tools. The target execution hardware platforms can be either homogeneous or heterogeneous while considering general embedded processors and hardware accelerators implemented with reconfigurable logic fabrics. Dynamic power management techniques are also considered in the design flow for runtime power optimisation.

To keep a smooth transitions between ideas while targeting next generation self-embedded systems, first we will expose the contribution of dynamic execution model for avionic simulation and test as it covers high performance domain. After that, we will detail the contributions in the fields of the low power design and the massively parallel dynamically reconfigurable execution model successively. All the contributions will help for the definition of our future project as it will be detailed in Chapter 5.

1.3.1 CPU/FPGA dynamic execution model for avionic simulation and test

The ever growing competitiveness in the aerospace industry, pushes avionic stakeholders to revisit and strengthen their methodology and tools for the Verification and Validation (V&V) design process. In recent years, the feasibility of using reconfigurable hardware is being explored in the field of avionic, aerospace and defence applications [147] [115] [71] [134]. However, using FPGAs in such applications has its own challenges since time, space, power consumption, reliability and data integrity are highly crucial factors. However, there is no a coherent design process that explicitly details the V&V of the reconfigurable hardware through the different phases : simulation, test and integration.

Addressing the above challenge, we started in the last quarter of 2009 studying the development of new design process based on cutting-edge technology. The objective of this process is to bring reliability and competitiveness to the avionic industry. In this context, *i)* we advocated for a reconfigurable-centric design process dedicated to avionic systems considering all the design steps. Along this process, we redefined the role of the FPGA circuit to cover the simulation, the test and the integration steps. First, reconfigurable logic is used in the frame of heterogeneous CPU/FPGA computing in order to obtain fast real-time simulation. Second, the FPGA is used as a key solution to offer versatile test benches and to converge toward unified test and simulation tools. Third, at the integration phase, we meet the conventional tools to make profit from reconfigurable technology in embedded avionic applications in order to deliver high computation rates and to adapt their functioning mode to provide reliability, fault tolerance, deterministic timing guarantees, and power efficiency [4] [6]. *ii)* We defined a generic and scalable heterogeneous CPU/FPGA environment as well as the corresponding dynamic execution model to bring self-reconfiguration to the system. Two international patents describing the innovative system for avionic simu-

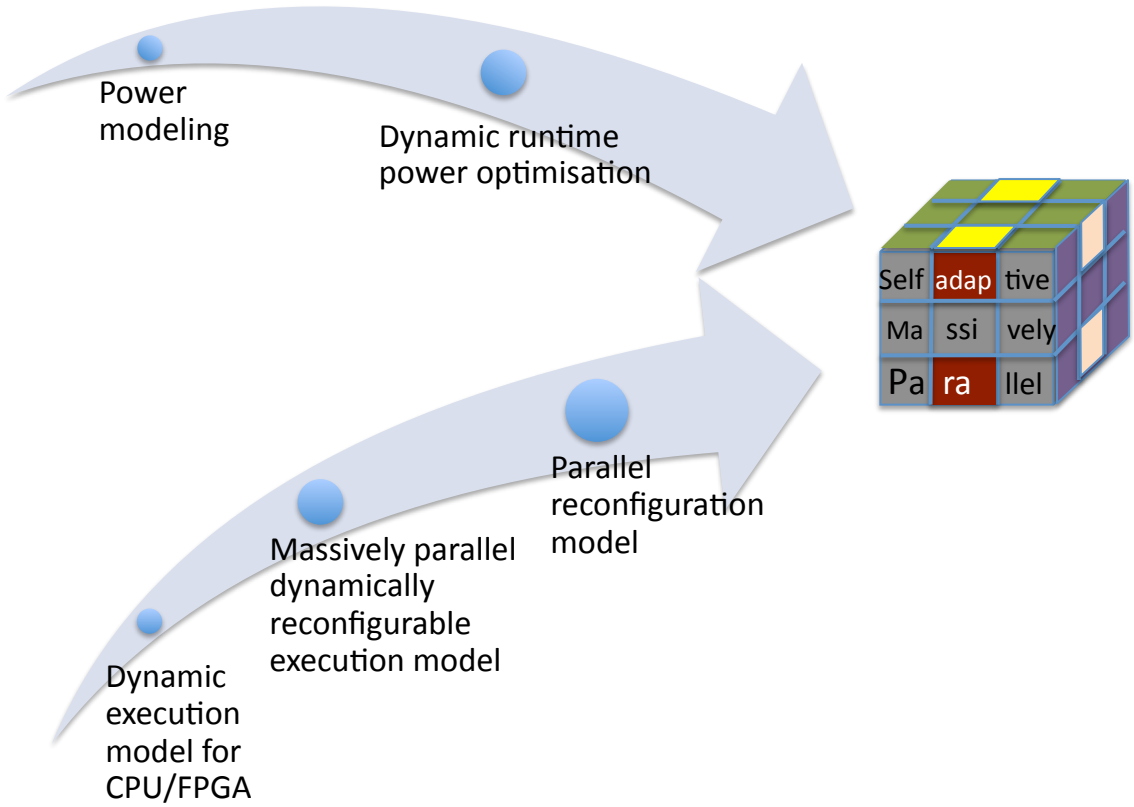

FIGURE 1.1 – *Contributions presented in this work*

lation and test are registered at the INPI ("Institut National de la Propriété Industrielle") and at the Australian Patent Office in collaboration with Airbus Helicopters. *iii)* We investigated the problem of the optimisation of run-time task mapping on a real-time computing system CPU/FPGA used to implement intimately coupled hardware and software models. This work includes the development and the comparison of mathematical models that focus on the static initial task mapping and efficient heuristics for the dynamic mapping of new applications at run-time, and the dynamic reconfiguration to avoid the real-time constraint violation [36] [35]. *iv)* We developed a software real-time simulation environment running on a heterogeneous CPU/FPGA system [13] [7] [2].

The target systems (aircraft, helicopters, etc.) are very complex and they are considered as System-of-System (SoS). We already started studying the scalability of the environment to construct a network of heterogeneous computing nodes where the reconfigurable technology will play an essential role. This led to a new patent registration at the INPI [8] in collaboration with Airbus Group. In the future, we will pursue the research on dynamic execution model considering distributed and heterogeneous systems.

A reconfigurable technology-centric design process

Targeting the objective of a unified and versatile environment for simulation, test and integration, we started with the definition of the main requirements of such environment [17].

In fact, the system should be generic to support any helicopter range or avionic equipment, scalable regarding the number of the computing nodes or the communication interfaces, adaptive to associate the appropriate models for a given scenario, and dynamic to be reconfigured during the system runtime.

In order to satisfy the above requirements, we will rely on reconfigurable technology as an essential part of our environment for many reasons. For the first aspect, nowadays reconfigurable circuits such as FPGAs can host different computing nodes such as hard-cores, soft-cores and hardware accelerators. Furthermore, it can be coupled with other computing nodes such as General Purpose Processor (GPP) and interfaced with a widespread communication standards. FPGA can answer to the dynamicity requirement through the DPR feature. The advantages of using FPGAs in the development of avionic systems are transverse to the design phases [17].

- At an early phase, we involved the reconfigurable technology in the design process for real-time simulation. We proposed the usage of FPGAs to design heterogeneous CPU/FPGA architecture that could implement intimately-coupled hardware and software avionic models. The main objective is to deliver high performance computing with real-time support. FPGA brings also dynamic reconfiguration capability to the system in order to deal with runtime model re-allocation. Furthermore, this step allows to verify the eligibility of a given model to be implemented as a cost-effective hardware solution compared to a software implementation [31] [32].
- As a transition between the simulation and test phases, we propose first to use the FPGA as a bridge between virtual models and avionic equipments in the loop. At this level, reconfigurable technology is a key solution for the avionic I/O hardware obsolescence issue taking into consideration communication protocols as IPs [40] [41] [18]. The huge logic budget available in nowadays FPGAs allows to use these circuits for computation as well as for communication at the same time. Furthermore, we will support dynamic behaviour in order to switch between a simulated model to the real equipment or to switch between different avionic protocols [17].
- For the integration phase, we will rely on a standalone FPGA-based technology in order to carry out the avionic functionality. At this level, our concerns cover embedded constraint verification, fault tolerance, reliability, certification, etc.

The dynamic CPU/FPGA hardware environment

we proposed a scalable heterogeneous CPU/FPGA hardware environment composed mainly of two nodes. The first node is a general purpose multi-core processor (i.e. : AMD/Intel) while the second node represents an FPGA. The multi-core will offer performance with a limited parallelism capability due to the fixed number of cores. FPGA is the support of the reconfigurable logics needed to implement challenging models (or tasks) as hardware accelerators. Designers could exploit the existing partitioning in the application (i.e. hardware-software and parallel-sequential hardware) which leads to several feasible implementations whose performances vary with the chosen partitioning [5] [3]. Our expectation of the above described architecture is to prototype some models which can be eligible and relocated in the FPGA. The objective is to increase the performance of these models and to reduce the communication latencies by the means of embedding different parts in the same chip. Within our environment, a great care has been devoted to the real-time aspect in order to satisfy tight computing and communication deadlines related with the target application domain

(soft real-time constraints) [4] [6].

For the dynamic execution model, each avionic model can be designed with different versions (i.e. software, hardware, etc.). A common high level model is developed in order to include different functions which correspond to different implementations. The necessary data (input, output, current context) is contained in a global data structure stored in the shared memory allowing easier context switch from a software node to a hardware node and vice-versa at runtime and without a full simulation restart. As an essential functionality of our environment, we can anticipate overflows, take the decision to reconfigure, run a heuristic for rapid mapping solution depending on the available nodes, and finally reconfigure the system [2] [49].

Mapping and scheduling of tasks on CPU/FPGA system :

The usage of CPU/FPGA architecture in the context of simulation and test environment needs tools to map efficiently tasks on the heterogeneous computing nodes. As all the connections between the different nodes are allowed, the communication delays are also heterogeneous. Targeting the initial mapping, we focused on the mathematical modeling of a scheduling problem in a heterogeneous CPU/FPGA architecture with heterogeneous communication delays in order to minimize the makespan, C_{max}. This study was motivated by the quality of the available solvers for Mixed Integer Program (MIP). The proposed model includes the communication delays constraints in a heterogeneous case, depending on both tasks and computing units. These constraints are linearised without adding any extra variables and the obtained linear model is reduced to speed-up the solving with CPLEX up to 60 times. The computational results show that the proposed model is promising. For an average sized problem up to 50 tasks and 5 computing units the solving time under CPLEX is about few seconds which is a reasonable time for the initial mapping of the system [9] [11]. We highlighted that the particular case of homogeneous multiprocessor scheduling with heterogeneous communication delays has been already resolved in [10]. Actually, we proposed a new MIP formulation that drastically reduces both the number of variables and the number of constraints, when compared to the best mathematical programming formulations from the literature [10].

Our investigation concerned also the development and the comparison of efficient heuristics that focus on the dynamic mapping of new applications at run-time, and the dynamic reconfiguration to avoid the real-time constraint violation. The Greedy heuristic LPT-Rule (Longest processing Time Rule), HEFT (Heterogeneous Earliest-Finish Time) Heuristics are explored in our work. Compared to an exact methods, these heuristics offer a good optimality in a time magnitude of milliseconds [36] [35].

Real-time simulation environment :

During the manufacturing process, designers need development tools for the verification and the validation of modern complex systems. Today, the simulation phase is considered as an unavoidable part of the V&V cycle. In order to meet the application requirements in terms of increasing computation rate and real-time, dedicated simulators should be used. Simulation tools should make profit from nowadays high performance architectures. Previously, we emphasized the usage of heterogeneous multi-core CPU/FPGA as an efficient

execution support for real-time simulators. However, there is a lack of real-time simulation environments able to deal with the execution of applications on such heterogeneous systems.

We investigated the development of soft real-time simulation environment supporting CPU/FPGA hardware architecture [13] [7] [2]. In such environment, we exploit the available hardware resources for the dynamic task switching between multi-core CPU and FPGA. The main features of our environment are the following :

- Creation and launching of a graph of tasks composed of hardware and software models on the available heterogeneous resources,
- Synchronization and communication between hardware and software models,
- Real-time monitoring of the available computing resources,
- Supervision of a simulation project in order to detect violation of timing constraints and to anticipate overflows and reconfigure the system at runtime.

1.3.2 Energy/power-aware design for homogeneous and heterogeneous MPSoC :

Due to the growing computation rates of nowadays embedded applications, using Multiprocessor System-on-Chip (MPSoC) becomes an incontrovertible solution to meet the functional requirements. In such systems, power/energy consumption is a critical pre-design metric that should be considered in the design flow. In current industrial and academic practices, power estimation using low-level CAD tools is still widely adopted, which is clearly not suited to manage the complexity of embedded systems supporting modern applications. In fact, MPSoCs have a huge solution space at the application, the Operating System (OS), and the architectural levels, which makes the Design Space Exploration (DSE) complex. This challenge is addressed by several frameworks through the development of Electronic System Level (ESL) tools. The objective is to unify the hardware and software design and to offer a rapid system level prototyping using virtual platforms. Based on the design step and the requirements like the timing accuracy and the estimation speed, designers could select an appropriate abstraction level to model the software simulating the system. Unfortunately, most of existing tools do not consider the power metric or focus on power estimation for a given abstraction level without overcoming the wall of speed/accuracy trade-off.

To answer the above described challenge, we propose the following contributions : *i)* an energy/power-aware design methodology for embedded applications executed on MPSoC [16] is proposed. It is based on a multi-level design flow in order to evaluate and to optimise the energy/power on the base of complementary models of hardware and software components. Our design methodology focuses on the functional and the transactional levels to deal with the design complexity and the broadness of the architectural solution space. *ii)* A DSE strategy is defined to refine the solution space while switching between the abstraction levels. This exploration step includes runtime optimisation techniques that are developed and integrated in the design methodology to reduce energy/power consumption of the system [24]. *iii)* Functional-Level Power Analysis (FLPA) is used to elaborate different power models (processors, hardware accelerators, OS services, etc.) that are plugged afterwards in our tools at the different abstraction levels to evaluate the total consumption of the system. *iv)* Model Driven Engineering (MDE) is used to automatise the design process and the plug-in of power models [39].

In the future, we will pursue this research direction considering the fact that IC 2D scaling is reaching the fundamental limits. Hence, the semiconductor actors are exploring the use of the vertical dimension (3D) for logic and memory devices. The combination of 3D device

and low power device will introduce a new era of scaling, identified in short as *3D Power Scaling* [83]. Furthermore, thermal effects are exacerbated in 3D technology. So, we need to rethink about the power management for the next generation of 3D-based multiprocessor SoC.

Power-aware design methodology :

In the frame of the OpenPeople project [34] [33], we proposed a multi-level power-aware design methodology for MPSoC that covers several design layers. The objective is to offer a power estimation tool for each step in order to have a gradual refinement of the design space solution based on the power or energy criteria. In order to cope with the design complexity, we focus specially on the functional and the transactional levels that offer different trade-offs between accuracy and estimation time. For each level, several models are developed for estimating and optimising the power consumption taking into account all the embedded system relevant aspects ; the software, the hardware, and the operating system. In this work, we based on the same power modeling approach (FLPA) for the the functional and transactional levels in order to guarantee the coherence of the estimation strategy in our design methodology. Our methodology helps designers to plug these power models with the design tools, to explore new architectures, and to apply optimisation techniques in order to reduce energy and power consumption of the system [16] [26].

The multi-level design space exploration :

Multi-level DSE is an unavoidable solution to have a good speed/accuracy trade-off. Actually, a top-down DSE allows fast to eliminate the undesirable solutions at each design level before reaching physical implementation levels. In the frame of multi-level DSE, designers of embedded applications need a seamless power-aware design methodology that takes into account the power metric at different abstraction levels. In a top-down design methodology, an appropriate power estimation and optimisation tool should be defined according to each abstraction level. The objective is to offer a gradual refinement of the solution space while switching between the design steps. Leveraging high abstraction levels is certainly the key ingredient for reaching this objective. Indeed, higher level approaches are fast, cost-effective and reliable enough to compare different architectural solutions. Using virtual platforms according to the abstraction level is inevitable in order to collect the strict relevant data (the values of the power model parameters) depending on the design step. During several years, we explored different abstraction levels such as Cycle Accurate (CA) [15], TLM (Transaction Level Modeling) with Instruction Set Simulator (ISS) [29] or Just-In-Time (JIT) techniques [30] [100], functional [26], and abstract clock-based approach [1]. The step of DSE must identify the adequate parallelism level, the parameters configuration, the hardware/-software mapping, etc.

The power modeling approach :

At the system level, we need power models emulating the behaviour of the different parts of the system in terms of consumption. The power modeling process is centred around two correlated aspects : the power model granularity and the main activity characterisation. The main challenge is to define a generic power modeling approach that can cover

the different abstraction levels and guarantee the coherence of the estimation strategy for a seamless power-aware design methodology.

In our work, the Functional-Level Power Analysis (FLPA) is used to develop generic power models for different target platforms. FLPA comes with few consumption laws, which are associated to the consumption activity values of the main functional blocks of the system. Basically, the FLPA is used for processor power modeling. In our research, it was extended to cover the other hardware components used in the MPSoC such as the memory, OS services and the reconfigurable logic. In the energy analysis step, various hardware and software parameters which influence the energy consumption are identified and then energy profiles are traced according to the variation of these parameters. From the energy traces, a curve fitting will allow us to determine the power consumption models by regression. The obtained power models are expressed in the form of analytical equations or table of values. The proposed approach aimed to extract power/energy models of embedded OS services, software application and hardware components. The generated power models have been adapted to system level design, as the required activities can be obtained from a system level environment. In our case, power models are plugged into our design tools at the functional and transactional levels. This approach was proven to be fast and precise [16]. The main advantage of this methodology is to obtain models which rely on the functional parameters of the system with a reduced number of experiments.

Model driven engineering :

As a main result of my first Post-doc, we developed a Model Driven Engineering (MDE) based environment for MPSoC design [21]. We pursued the investigation for the usage of MDE to model power consumption aspects and to automatise the plug-in of power models in the design process [39]. Indeed, MDE is needed in order to make the SoC design easy and not tedious, by making the low-level technical details transparent to the designers. In MDE, models become a mean of productivity.

The main contribution in this field was a hybrid energy estimation approach for SoC, in which the consumption of both white-box IPs and black-box IPs can be estimated. We highlight that white-box refers to open-source IP while black-box concerns Proprietary IP. Based on MDE, this approach allows to take the consumption criterion into account early in the design flow, during the co-simulation of SoC. In a previous work [14], we presented an annotated power model estimation technique for white-box IPs where counters are introduced into the code of the IPs. A counter is incremented whenever its related activity occurs as described in [20] [19]. This technique was used in this work, along with the standalone power estimator technique used for black-box IPs. The standalone power estimation modules were generated using MDE and connected between the components in order to detect their activities through the signals that they exchange. To test this approach, systems containing white-box IPs and black-box IPs and their related estimation modules were modeled in the Gaspard2 framework. Using the MDE model transformations, the code required for simulation can be generated automatically. Finally, power consumption estimates can be obtained during simulations.

1.3.3 Massively parallel dynamically reconfigurable execution model

Standard Integrated Circuits (IC) are reaching their limits and need to be extended to meet the next-generation computing requirements. One of the most promising evolutions is 3D-Stacked Integrated Circuits (3D SICs). Recently, SICs technology, also known as 2.5D ICs, has been released by the manufacturer Xilinx for the Virtex 7 FPGA family. Such technology is considered as a near-cousin to 3D. The next-generation 3D FPGAs (three-dimensional Field Programmable Gate Arrays) will allow efficient dynamic reconfigurations in a massively parallel manner. According to their needs, software applications running on such hardware can then efficiently reconfigure the hardware at runtime, thereby achieving significant savings in circuit space, energy consumption, and execution time [82].

We believe that 3D integration will lead to a significant shift in the design of FPGA circuits. Indeed, by incorporating the configuration memory on the top of the FPGA fabric, with fast and numerous connections between memory and elementary logic blocks, it will be possible to obtain dynamically reconfigurable computing platforms with a very high reconfiguration rate. This opens the possibility of creating massively parallel IP-based machines. Such architectures can be customized at runtime using the DPR feature, a reconfiguration that can be done in parallel for all or for a subset of the IPs. This new hardware paradigm opens many opportunities for research since there are no parallel reconfiguration models for such technology, no execution models for massively parallel and dynamically reconfigurable architectures on 3D FPGA, and no dedicated tools for mapping those architectures on 3D FPGA or estimating their performances.

To overcome the above-mentioned obstacles, *i)* We conceived a Multi-FPGA board as an appropriate execution support for massively parallel and dynamically reconfigurable architectures. *ii)* We implemented a proposed parallel reconfiguration model that takes profit from the innovative 3D technology to allow fast and simultaneous programming of several logic fabric regions. This parallel reconfiguration model was emulated on the Multi-FPGA board. *iii)* We defined an efficient execution model dedicated for massively parallel and dynamically reconfigurable architectures. This execution model has been implemented through the HoMade processor.

3D packaging is the next innovative technology for FPGAs. The inter- and intra-layer positioning of communication and logic resources is of utmost importance. We anticipate that multiple stacked layers can be used for a fast and massively parallel reconfiguration over the whole chip as confirmed in [126]. As soon as 3D FPGAs become available, our future works on Multi-HoMade execution model could be deployed on 2D Multi-FPGA platform taking benefits of all previous results.

Multi-FPGA System-in-Package :

As we are waiting for 3D packaging in the next FPGA generation, the first validation of massively parallel dynamically reconfigurable architecture is performed using currently available Xilinx FPGAs which are not currently supporting parallel reconfiguration. In parallel, we proposed to use an emulation platform in order to implement massively parallel and dynamic architectures and to reconfigure several cores/regions in parallel. Recently, we designed in collaboration with the Nolam Embedded Systems [10] a multi-FPGA board featuring a parallel reconfiguration mechanism. The main idea is to have a parallel reconfigurable

10. www.nolam.com

architecture that also provides modular technology with customizable and reconfigurable computing power. The application domain includes a wide range of sophisticated applications with a specific focus on intensive signal processing applications used in the avionic domain. In order to provide high performance and dynamicity capabilities, the board provides two main features : parallel runtime reconfiguration and peer-to-peer high-bandwidth low-latency communication link which are implemented using a PCIe Gen3 switch on-board.

Our board has a parallel I/O management model using the FPGA Mezzanine Card (FMC) and PCIe switch. Each FPGA can communicate with the outer world with an FMC module. However, in case of distributed processing, the data received via a single FMC might need to be shared with more than one FPGA. In this case, the owner of FMC I/O can share its data with more than one node at the same time via the PCIe switch. We can also reconfigure our board as a parallel reconfigurable machine with a shared memory model. Each FPGA has a local DDR3 memory while the master FPGA has a memory size four times compared to the local one. Each FPGA can store its complete local memory in the global shared memory via the PCIe switch. The Master FPGA in turn can retransmit the data to one or more nodes at the same time if requested thus forming the notion of a global shared memory.

The prototype of the board is realized with a carrier board and 4 FPGA modules. FPGA modules contain only the FPGA and the need electrical circuitry to support the operation of the FPGA. The connector on the FPGA module is used to mate with the carrier board. The size of the FPGA module has been chosen taking into consideration the size of the largest FPGA device in the market. On the other hand, the carrier board consists of all the other components and features (i.e., FMC, memory, PCIe switch, COM express and peripheral I/O interfaces) of the multi-FPGA board.

Parallel reconfiguration model :

The key architectural feature of the Multi-FPGA board is to be able to reconfigure more than one node at the same time. Using this feature, we aim to emulate a parallel reconfiguration model of 3D FPGAs respecting the Single Program Multiple Data execution model (SPMD). In a SPMD architecture, where multiple instances of the same IP process different data sets, we should reconfigure several IPs or a subset of IPs when the context of the application changes. In practical terms, it will not be efficient when reconfiguration is done sequentially for a large number of IPs, using current generation 2D FPGA based reconfiguration model. Since current 3D FPGAs are still emerging as an inevitable technology, we still need to speculate the partial reconfiguration possibilities that these devices will offer for such high-density applications and emulate the behavior of such a reconfiguration model with current generation FPGAs. Based on this premise, we propose a partial reconfiguration model for next generation 3D FPGAs well-traced on the execution model (SPMD) in order to reconfigure in parallel a subset of the computing nodes. To validate our approach, we rely on a multi-FPGA based architecture that can support parallel communication capabilities with two or more FPGAs at the same time.

HoMade processor :

As stated before, there is a need of a massively parallel dynamically reconfigurable execution model to deal with the high requirements of sophisticated embedded systems. Unfor-

tunately, to the best of our knowledge, there is no processor supporting such execution model or proposing a programming language for this application domain. Since the last quarter of 2011, we have devoted significant efforts inside the DreamPal INRIA team to address this challenge. This has led to the development of the ***HoMade processor***.

While programming an FPGA, hardware designers generally tend to use two approaches. The most commonly used is to develop hardware accelerators on FPGAs, which are dedicated circuits that are an order of magnitude faster than a software implementation. The other approach consists of using softcore processors which are processors implemented using hardware synthesis. They can be proprietary solutions such as MicroBlaze from Xilinx, Nios from Altera, etc. or open-source solutions like Leon, OpenRISC, FC16, etc. In our previous work, we already used such solution for implementing a driver assistant system on Xilinx FPGA [23] using a Microblaze. The hardware solution can be adapted at runtime according to the environment parameters [22]. The choice of an appropriate softcore processor for a given application is wide and many new solutions emerge, including multi-softcore implementations on FPGAs. In [25], we conceived a Multi MIPS parallel architecture around a multistage interconnection network on FPGA. Between these two approaches, there are other various approaches that connect IPs to softcores, in which, the processor's machine code language is extended, and IP invocations become new instructions.

Based on this concept, we propose the HoMade processor that goes further by generalizing the notion of IP where ***everything is implemented in IPs*** ; only the control flow is assigned to the softcore itself. The partial dynamic reconfiguration of nowadays FPGAs makes possible such dynamic IP management in practice. ***HoMade is a reflective softcore processor*** that means it is able to modify its own structure and behaviour while it is running. Thus, HoMade can dynamically add, remove, and replace IPs in the application running on it. Also, HoMade is able to dynamically modify its own program memory, thereby dynamically altering the running program. We believe that efficient reflective softcores on the new 3D FPGAs should be small ; low performance generic hardware components (ALU, registers, memory, I/O...) should be replaced by dedicated high performance IPs.

Parallel execution is performed on a set of HoMade slave processors. It is handled inside HoMade as an SPMD model executed on slave HoMade processors. First, all the selected slaves run a program with a common start address. Second, a synchronisation barrier detects the end of the processing of a subset of the slaves. Controlled by an HoMade master, we carry out a program on a selected subset of slaves following the SPMD execution model. Between disjoint subsets of slaves, we propose a multi SPMD execution, each softcore subset running in a local SPMD mode. A substantial effort has to be done to support a large number of slaves without degrading the clock frequency. So, we need to manage activity of each slave. Again, an IP will be proposed by default to manage the activity locally on each slave and globally on the master. Users can redefine their own activity managers and can change the IP during the execution if needed. Parallel computing requires inter-processor communication. In line with the HoMade philosophy, communications between slaves will be implemented via specific IPs. Hardware designers can use the default IPs provided with HoMade which is a Network-on-Chip (NoC) 2D grids or develop their own IPs and even change the communication system at runtime thanks to dynamic reconfiguration.

Dynamic reconfiguration management will be itself performed by dedicated IPs. An ICAP like (Internal Configuration Access Port) primitive will be encapsulated within an IP. Using an IP to perform reconfiguration gives us more flexibility for performing reconfiguration, in particular, with respect to how and by whom reconfiguration may be triggered. Of

course, the targeted FPGA should support this functionality. A patent is under registration to highlight the HoMade execution model.

1.4 Outline

The remainder of the document is organized as follows : Chapter 2 summarizes our contributions on dynamic reconfigurable systems for avionic simulation and test ; Chapter 3 presents our works on power-aware design methodology of homogeneous and heterogeneous MPSoC ; Chapter 4 reports our recent works in the field of massively parallel dynamically reconfigurable execution model ; finally, Chapter 5 gives the conclusions and draws our future research directions. A complete Curriculum Vitae, and a list of publications are also provided in Appendix A.

Chapter 2

CPU/FPGA Dynamic Execution Model for Avionic Simulation and Test

This chapter presents my contributions since 2010 in LAMIH laboratory and INRIA DaRT team in the field of dynamic reconfigurable high performance systems. Mainly, these works were achieved in the frame of industrial collaborations with Airbus Group (previously EADS), Airbus Helicopters, and Nolam Embedded Systems. This work covers the PhD thesis of George Afonso started in August 2010 and defended in July 2013 (co-advised by Jean-Luc Dekeyser) for the CPU/FPGA hardware part, the PhD of Omar Souissi (October 2011 - December 2014) and the Post-doc fellowship of Abdessamad Ait El Cadi (January 2013 – December 2014) for the mapping and scheduling part, and the research internship of Zeineb Baklouti (April 2012 - February 2014) for the real-time simulation environment.

The remainder of the chapter is organized as follows : Section 2.1 introduces the main challenges for for avionic simulation and test ; Section 2.2 gives a summary of the related works. Section 2.3 introduces the essential of simulation and test avionic domains and exposes the proposed reconfigurable-centric avionic design process. In Section 2.4, our solution of reconfigurable computing for simulation is exposed. Section 2.5 presents an FPGA-centric solution for test systems. Technological issues and solutions for embedding avionic applications based on reconfigurable technology are enumerated in Section 2.6. To evaluate our approach, experimental results are presented in Section 2.7 through several case-studies. Finally, in Section 2.8, we discuss the strengths, limitations and future directions to the presented works.

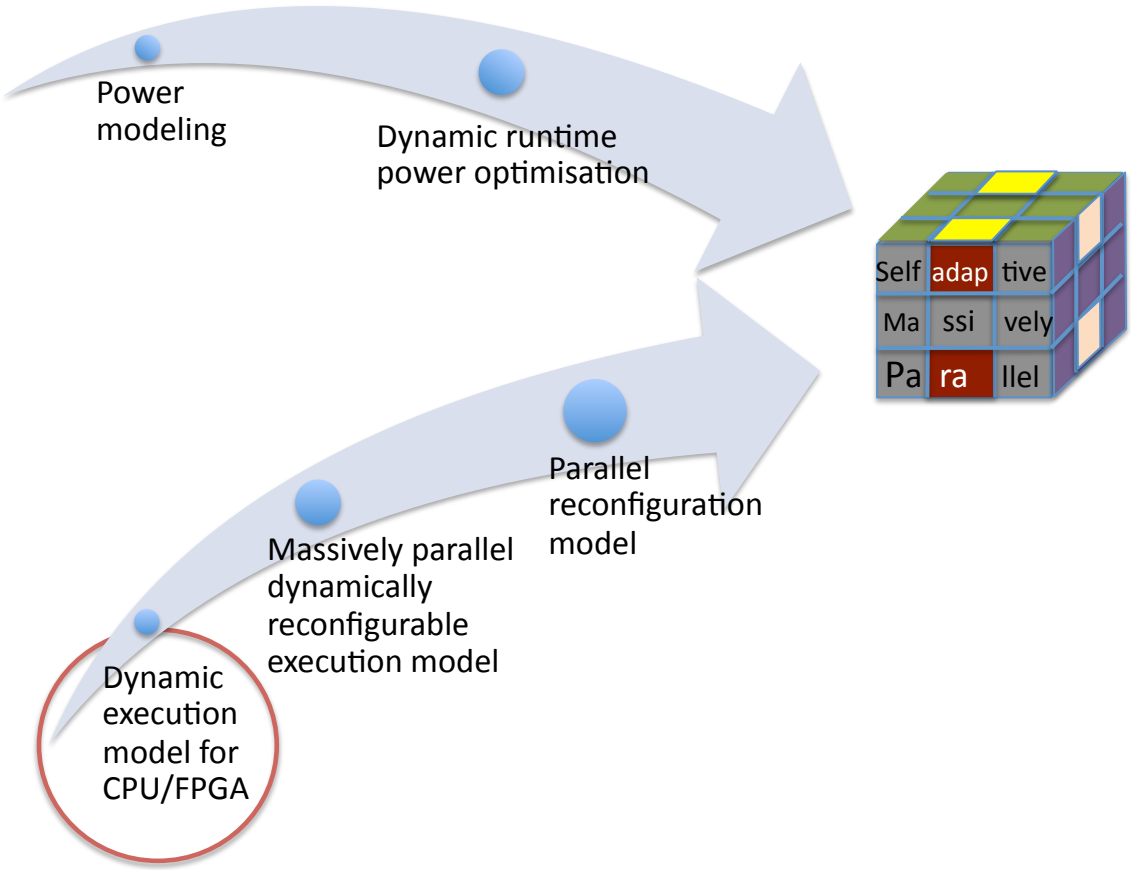

FIGURE 2.1 – Contributions the field of dynamic reconfigurable high performance systems

2.1 Main challenges for avionic simulation and test

Continuously growing aerospace industry competitiveness pushes avionic actors to revisit and strengthen their methodology and tools of the Verification & Validation (V&V) design cycle. In this perspective, the technical areas of simulation, test and integration systems are currently in an unavoidable convergence path. Yesterday considered as different expertise fields, these design steps should rely on common frameworks in the future supported by cutting-edge hardware architectures.

As examples of target avionic systems, we quote control, collision avoidance, pilot assistance, target tracking, navigation and communications, amongst other functions. According to the characteristics of these functionalities, high computation rates should be well-delivered while carrying-out intensive signal processing. Furthermore, these embedded systems often operate in uncertain environments. They should adapt their functioning mode to provide reliability, fault tolerance, deterministic timing guarantees, and energy efficiency. Undoubtedly, the essential feature of systems to reconfigure themselves (at the hardware or the software level) at run-time comes with additional complexity in the different design cycle steps. Along this work, we will take the Airbus Helicopters [11] avionic development cycle as an example. In the present industrial practices, different simulation tools and test benches are used for the verification of embedded avionic equipments (automatic pilot, navigation, etc.) dedicated to various helicopter ranges. This methodology calls for separate teams with different domain experts in order to achieve the simulation, the test and the integration of each part. Today, this process is very complex and expensive to perform. Actually, there is an essential need of a seamless process that could help designers during the V&V cycle starting from a full software simulation to the integration phase.

In parallel, FPGA reconfigurable circuits have emerged as a privileged target platform to implement intensive signal processing applications. FPGAs offer inexpensive and fast programmable hardware on some of the most advanced fabrication processes. FPGA technology can embed parallel hardware components or several IPs (Intellectual Property) due to the large number of programmable logic fabrics available on the chip. Such architectures can be customized at runtime using the Dynamic Partial Reconfiguration (DPR) feature, a reconfiguration that can be done for all or for a subset of the IPs. Academic and industrial expectation promote the idea that adaptive architectures will dominate next-generation embedded systems, including those based on FPGAs. We are in line with this vision, indeed this new hardware paradigm opens many opportunities for research in aerospace and avionic industries since there are no standard process to take into consideration the FPGA as an essential part of the design process starting from a full simulation to the integration phase.

In this perspective, it is necessary to consolidate the usual design cycle with a solid system approach allowing to early check/validate the adequacy and the consistency of the reconfigurable technology to be embedded in the aircraft. Such a system approach can be conceived only if the means of V&V are present upstream. In another meaning there is a seamless process to simulate as soon as possible the solution relying on reconfigurable technology and after that testing on benches before embedding in the real-world.

To overcome the above-mentioned challenges, we proposed a reconfigurable-centric design process dedicated to avionic systems considering the following steps :

11. Airbus Helicopters is the leader in civil and military helicopter manufacturing. http ://www.airbushelicopters.com

– First, for the simulation phase, we propose the usage of the reconfigurable technology in the frame of generic and heterogeneous CPU/FPGA architecture that could implement intimately-coupled hardware and software tasks. In our proposal, a great care has been devoted to deliver high performance computing, real-time support, and dynamic reconfiguration capability.
– Second, for the test phase, we propose a modular, runtime reconfigurable, and IP-based approach for the avionic communication support. This support allows to manage dynamically different avionic communication protocols in order to consider Unit(s)-Under-Test (UUTs) (automatic pilot, navigation, etc.) in the test loop. Our hardware support leads to the convergence of simulation and test tools.
– Third, for the integration phase, we discuss the main technological issues and industrial solutions for embedding FPGA based-avionic systems in the aircraft after the V&V from the previous phases taking into consideration different metrics such as reliability, timing constraints, power consumption, etc.

2.2 Related works

In recent years, the feasibility of using reconfigurable hardware is being explored in the field of avionic, aerospace and defence applications [147], [115], [71], [134]. However, using FPGAs in such applications has its own challenges since time, space, power consumption, reliability and data integrity are highly crucial factors. Some of these challenges are being addressed at the technology level, and some of them at the architectural level. One of the main challenges of using reconfigurable hardware specifically in space missions is that, it has to be radiation and fault tolerant. Single Event Upsets (SEUs) are induced by radiation. The environment where the avionic systems operate has unfavourable effects in these devices. Therefore, it is important to provide a fault-tolerant computing platform for such applications which are prone to radiation effects. The works done by [134] and [101] address and mitigate the effects of SEUs on FPGAs and provide a reliable computing platform . Extensive work has been done in developing hardware/software co-design for an avionic communication system based on ARINC429 communication protocol [76]. Another related work also proposes the configuration and deployment of infrastructure and related procedures of a distributed avionic communication system in FPGA [101]. Such works serve as the foundation for the usage of FPGAs in avionic applications. However, there is no a coherent design process that explicitly details the V&V of the reconfigurable hardware through the different phases : simulation, test and integration.

Historically, tools used for avionic simulation and test have often been decoupled. This matter of fact could be explained by technical choices : Real-Time Operating System (RTOS) competitiveness, hardware access, and models management capabilities. Due to the hard time-to-market requirement, practices have started to change during the last years. With the new technologies in the fields of hardware architecture and the emergence of virtualization solutions, aerospace actors are reconsidering their methodologies to verify and validate critical embedded systems. The result of this wide technological motion is the vital need to converge toward unified simulation and test tools.

For the past twenty years, the avionic test systems were based on real-time specific hardware architectures such as the well-spread VME CPU boards [60]. The VMEBus is particularly efficient to allow Input/Output (I/O) event management, multi-processing synchroni-

zation, and a transparent access to the different hardware resources. As a most of aeronautic firms, Airbus Helicopters has integrated the VMEBus as a standard backbone for the test benches of embedded helicopter systems. The proprietary test system named ARTIST is based on VME technology and the VxWorks real time Operating System (OS). These technologies have been used for all helicopter benches in order to validate the avionic equipments.

Due to the present performance requirement, an increase in the computation rates is needed, but it cannot be delivered by the VME CPU boards anymore. Furthermore, this solution is considered as an expensive maintainable technology. To overcome these drawbacks, Airbus Helicopters recently decided to move to a "half generation" test system based on high performance PC or workstation solution. Upcoming architectures are based on multi-core computer plugged with I/O boards to communicate with the equipments under test. Airbus Helicopters has selected the PEV1100 VME Bridge solution [58] [59] from the Swiss company IOxOS [12]. The PEV1100 allows a local host to interface with a VME64x bus using a PCI Express (PCIe) external cable which offers transparent access to I/O boards. To achieve higher communication performances, IOxOS Technology had developed a dedicated interface between the PCIe and the VME64x bus. This interface is built with the latest FPGA technology in order to implement PCIe end-point hardware cores.

The usage of multicore hosts allows an immediate increase in the capacity of computation. An important outcome of this transition is the refusal of the obsolete CPU boards. However, this solution cannot guarantee the real-time criteria while the execution of concurrent tasks due to the lack of an appropriate OS environment. Furthermore, this solution brings new communication latencies between the CPUs and I/O boards plugged in the VME backplane.

Among existing avionic test systems provided by cutting-edge firms, we quote Aidass family [120] used in particular for Eurofighter Airbus Military Air Systems, U-Test [59] developed by EADS Test&Services, and ADS2 from Techsat GmbH [13]. The proposed solutions are fully based on CPUs resources (PC or VME boards) and are close to Airbus Helicopters's solutions. These test systems can both deal with I/O management and simulation environments. Today, the management of increasing computing power relies on additional CPUs. For simulation dedicated tools, Airbus Helicopters's internal solution RISE (Real Time Simulation Environment) described in [50] does not support reconfigurable resources for virtual models management. Our perspective in this project is to consider the FPGA as an essential sub-part of simulation, test, and embedded system architectures.

In addition, it has been proven that runtime reconfigurable hardware utilizes hardware resources much more efficiently. In [147], the authors propose a methodology for applications to be fault tolerant and sustain much longer using runtime reconfiguration capabilities. Using FPGAs for accelerating applications has shown significant performance improvements in aerospace applications [71], [134]. In this work, we used runtime reconfiguration in the frame of avionic design process to achieve real-time simulation with better performances, to converge between simulation and test domains and to conceive more adaptive and reliable avionic systems.

12. http ://www.ioxos.ch/
13. http ://www.techsat.com

2.3 Reconfigurable-centric avionic design process

2.3.1 Essential of simulation and test avionic domains

Avionic simulation and test (S&T) domains target the validation of avionic embedded systems before the first test flight in order to increase the safety and to reduce the time-to-market. These phases are critical and have to respect constraints in order to provide the duplication of real flight conditions. In order to perform a complete simulation or test session, we need to model each part of the helicopter and the environmental parameters (weather conditions, geographical factors, etc.). The simulation phase relies totally on virtual models. Figure 2.2 presents a simplified system simulation loop that simulates the helicopter behavior including three models : the *flight mechanic model*, the *navigation model* and the *automatic pilot model*. In the initialization phase, the flight mechanic model takes into consideration several parameters such as the initial position relative to the ground and the aircraft configuration file to give back an *equilibrium position*. In addition, it sends the *common data area* structure containing the position and the speed of the aircraft to the navigation model. This later computes the helicopter destination and sends it to the automatic pilot model via the *ordered roll* structure. Finally, the flight control is managed by the automatic pilot.

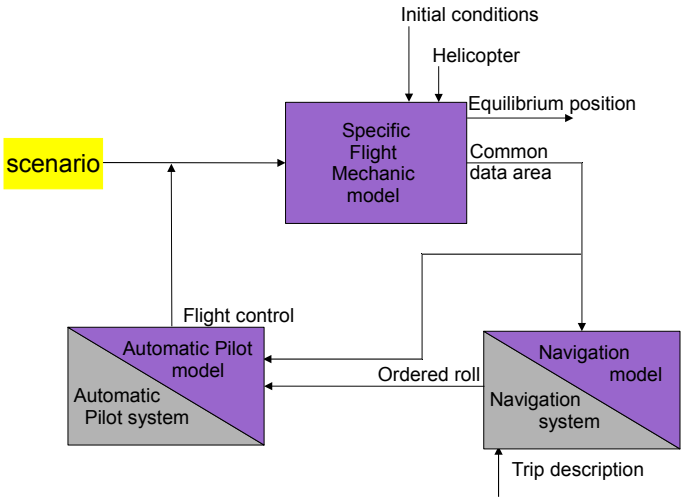

FIGURE 2.2 – A simplified simulation/test loop system

For the test scenario, each virtual model such as the automatic pilot can be replaced by the corresponding real avionic unit which calls for additional I/O communication hardware support. For example, the flight mechanic model can receive the *flight control* from a simulated model or from an I/O avionic interface (ARINC429, MIL-STD-1553, etc.) in the case of using a real automatic pilot system in the loop. These elements are essentials for the configuration of each test scenario depending on the Unit-Under-Test (UUT) and the timing constraints.

In the current industrial practices, the design cycle of a new avionic equipment is following the V diagram illustrated in figure 2.3. As a first development step, the specification of the system allows a preliminary study about the hardware architecture of the new avionic equipment at different levels. To do so, different simulation (virtual) models are developed offering a first environment for the pilots to interact with the new functionality. After a vir-

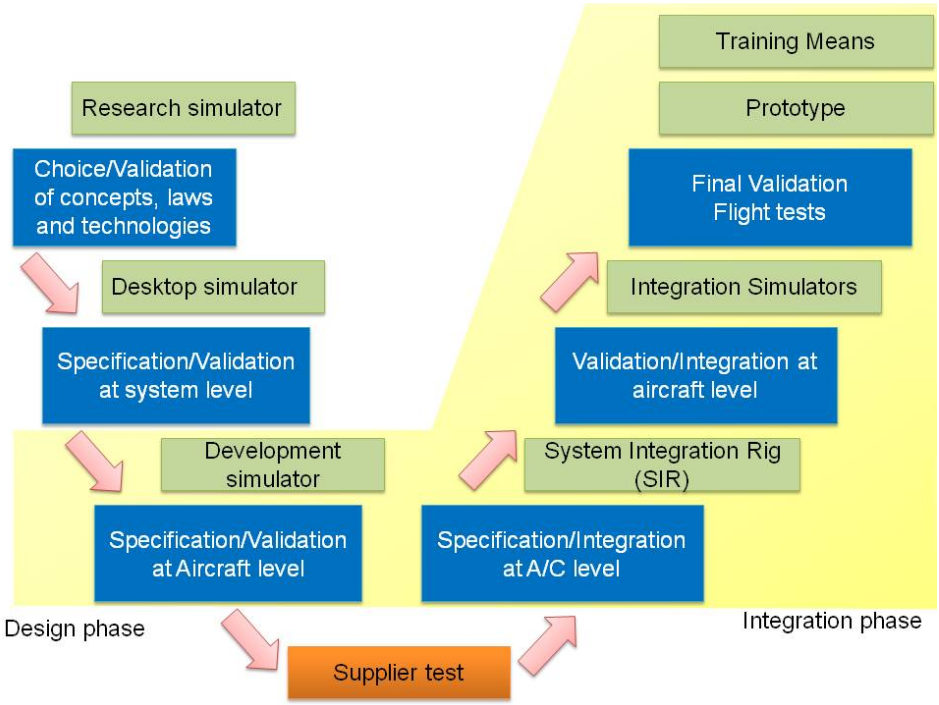

FIGURE 2.3 – New avionic equipment design cycle

tual validation, the system specifications are transmitted for design. At the integration phase, the equipment is validated first through test benches (System Integration Rig) before final flight tests.

Today, different test benches are used for the verification of various helicopter ranges and UUTs (automatic pilot, navigation, etc.). Each test bench relies on a specific hardware architecture. This is due to the heterogeneity of the helicopter parts in terms of computing requirements and handled data structures. In general, several specialised CPU boards are needed to satisfy real-time constraints which leads to sophisticated synchronization and communication schemes. In addition to this, dedicated avionic I/O boards (ARINC429, MIL-STD-1553, etc.) are required depending on the UUTs.

In conclusion, the presented design cycle calls for separate teams with different domain experts and several software tools in order to achieve each phase, hence this process is considered very complex and expensive to perform. Our objective is to converge toward a unified environment as shown in figure 2.3 with the yellow color. Our vision is centred around the reconfigurable technology that can play a key solution in such challenge.

2.3.2 The proposed design process

Face the above challenge, we started studying the development of new design process basing on cutting-edge technology. The objective of this process is to bring reliability and competitiveness to the avionic industry. In the last quarter of 2009, we started the development phase with defining the main features and characteristics of a namely *unified and versatile environment for simulation, test and integration* :

– Generic : the proposed environment should be generic in order to support any heli-

copter range or avionic equipment. The hardware architecture should follow the generic aspect of the environment in order to support various computation nodes, avionic communication protocols, etc.
- Scalable : The number of computing nodes or communication interfaces should be extensible according to the number of avionic systems.
- Adaptive : When a simulation, test or integration project is performed, the avionic models associated with a given helicopter range should be adapted according to the constraints (e.g. type, weight, size, etc.).
- Dynamic : At runtime, we can replace an avionic model with another or a communication protocol with a second. In addition, in the same environment we can switch between a simulation and test phases or vice-versa.

In order to satisfy the above requirements, we will rely on reconfigurable technology as an essential part of our environment for many reasons. For the first aspect, nowadays reconfigurable circuits such as FPGAs can host different computing nodes such as hard-cores, soft-cores or hardware accelerators. Furthermore, it can be coupled with other computing nodes such as General Purpose Processor (GPP) and interfaced with a widespread communication standards. For the scalability of the environment, FPGAs can be used to construct a network of computing nodes or parallel machines. In order to increase the productivity, FPGAs will be used in the frame of an IP-based design methodology promoting *All is IP* in order to favour the reuse and lead to more adaptive systems. In order to preform a given simulation, test or integration project, the user needs only to select the appropriate IPs (hardware or software avionic models, I/O avionic protocols, etc.) according to the constraints. With the Dynamic Partial Reconfiguration (DPR) feature, IPs can be managed at runtime to switch between different implementations and communication protocols.

These advantages of using FPGAs in the development of avionic systems are transverse to the design phases. As illustrated in figure 2.4, we redefine the role of the FPGA circuit to cover the simulation, the test and the integration steps. In what follows we detail the design process :

- At an early phase, we involve the reconfigurable technology in the design process for real-time simulation. The simulation phase is considered as an essential part of the industrial product manufacturing. In fact, it is required to validate the performance of complex equipments at an early phase through virtual models. For different avionic systems, specific real-time constraints should be fulfilled. This behaviour has to be validated first at the simulation level before integrating the functionality into the real system. We propose the usage of FPGAs to design heterogeneous CPU/FPGA architecture that could implement intimately-coupled hardware and software avionic models. The main objective is to deliver high performance computing with real-time support. FPGA brings also dynamic reconfiguration capability to the system in order to deal with runtime model re-allocation. Furthermore, this step allows to verify the eligibility of a given model to be implemented as a cost-effective hardware solution comparing to a software implementation.
- As a transition between the simulation and test phases, we propose first to use the FPGA as a bridge between virtual models and avionic equipments in the loop. At this level, reconfigurable technology is a key solution for the avionic I/O hardware obsolescence issue taking into consideration communication protocols as IPs. The huge logic budget available in nowadays FPGAs allows to use these circuits for computation as well as for communication at the same time. At this phase, there are also real-time

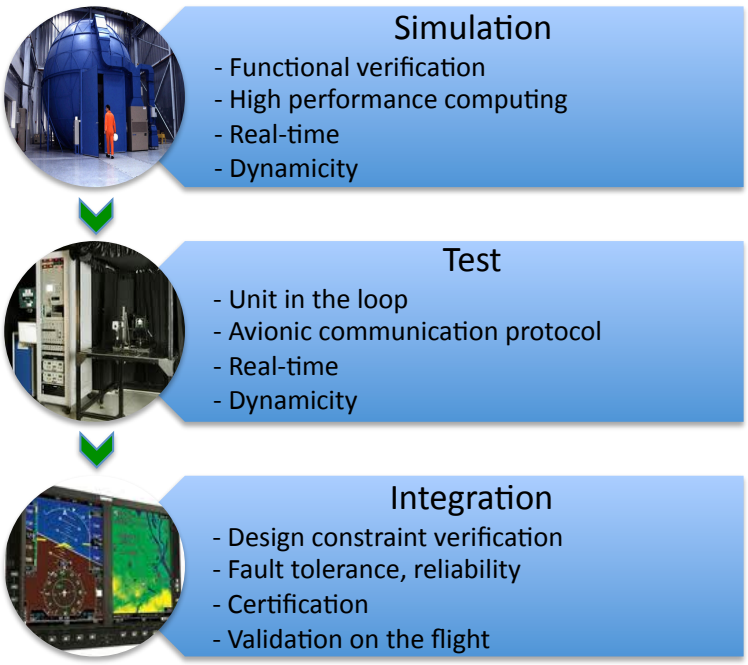

FIGURE 2.4 – The proposed design process

requirements with more complexity coming from the data synchronisation between virtual models and UUTs. Furthermore, we will support dynamic behaviour in order to switch between a simulated model to the real equipment or to switch between different avionic protocols. The test phase enables the interaction of the new functionality with existing avionic equipment before the integration phase.

– For the integration phase, we will rely on a standalone FPGA-based technology in order to carry out the avionic functionality. At this level, our concerns cover embedded constraint verification, fault tolerance, reliability, certification, etc.

2.4 Reconfigurable computing for simulation

As introduced in the previous section, the main goal of using reconfigurable computing for the simulation phase is achieving high computation rates with real-time capabilities. In order to meet these requirements, combination of general CPU and reconfigurable fabrics like FPGAs is necessary. In such systems, multi-core processors provide high computation rates while the reconfigurable logic offers high performance per watt and adaptability to the application constraints. Designers could exploit the existing partitioning in the application (i.e. hardware-software and parallel-sequential hardware) which leads to several feasible implementations whose performances vary with the chosen partitioning. With the management of the parallelism intrinsic in the application, FPGA technology could offer better performances comparing to CPUs or GPUs up to 10x [54] at lower frequencies. Using heterogeneous CPU/FPGA systems allows to adapt the architecture according to the application constraints and thus to optimize hardware resources. All these benefits stimulate system designers to redirect their efforts on reconfigurable computing for simulation domain.

Our expectation of the above described architecture is to prototype some models which can be eligible and relocated in the FPGA. The objective is to increase the performances of these models and to reduce the communication latencies by the means of embedding the different parts in the same chip. To do so, we need first to profile our avionic test loop in order to extract the complex models that will be implemented in the FPGA. Second, different hardware model configurations will be explored to reach an optimal well-balanced global system. Indeed, FPGA technology could implement heavy models in a hardware fashion with the management of the parallelism degree to address the real-time constraints of the application.

2.4.1 The heterogeneous CPU/FPGA hardware environment

As illustrated in figure 2.5, we propose a scalable heterogeneous CPU/FPGA hardware environment composed mainly of two nodes [4]. The first node is a general purpose multi-core processor (i.e. : AMD/Intel) while the second node represents an FPGA. The multi-core will offer performance with a limited parallelism capability due to the fixed number of cores. FPGA is the support of the reconfigurable logics needed to implement challenging avionic models as hardware accelerators.

Within our environment, a great care has been devoted to the real-time aspect in order to satisfy tight computing and communication deadlines. In fact, nowadays Operating Systems (OS) such as Linux allocate dynamically tasks onto the available cores which may introduce latencies and lead to the timing constraint violation. This is due to the fact that general purpose OS do not support real-time functionalities. Processor affinity service is a modification of the native central queue scheduling algorithm in a symmetric multiprocessing (SMP) operating system. Each task (process or thread) in the queue has a tag containing the target processor or core number on which it will be executed. In our architecture, we propose to allocate each kind of tasks (OS, avionic model, etc.) in the available cores under bounded soft real-time constraints. Figure 2.5 shows an example of task allocation ; cores 1 and 2 run the OS, core 3, 4 and 5 are dedicated to carry out the avionic models, the graphic part is mapped on cores 6 and 7, and finally the core 8 ensures the communication between the host and the FPGA module. For the reconfigurable part, several hardware models can be hosted in the FPGA while better performances are needed. such heterogeneous CPU/FPGA architecture could implement intimately-coupled hardware and software avionic models. The shared memory implemented in the software part allows data sharing between software and hardware avionic models.

As well as we need to optimise our avionic models in order to obtain better performances, we need also to focus on the communication which is crucial in heterogeneous architectures. The link has to be fast, efficient, and widely used in industrial systems. Nowadays, almost host machines or workstations are equipped with PCIe slots for expansion boards. In addition, a large range of commercial FPGAs integrate a hard endpoint PCIe core for industrial usage. Our proposal is to make profit from these features in order to design an efficient solution that can deal with the interoperability between hardware and software models mapped respectively on FPGA and CPU nodes with high throughput. In such architecture, communication latency with respect to the real-time constraints is considered the most important metric. Nevertheless, it is first necessary to define the application requirements in order to propose a customised solution that offers the better trade-off between the communication bandwidth and the design cost.

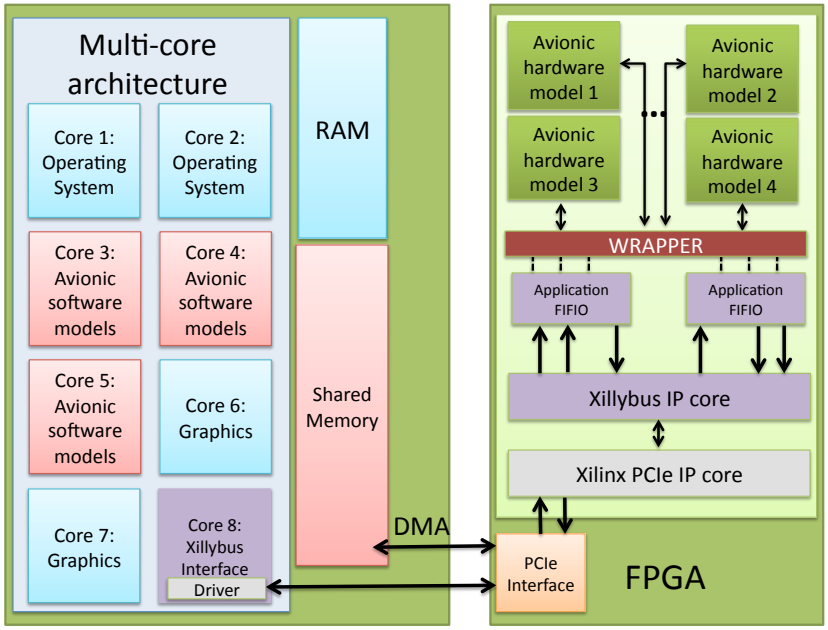

FIGURE 2.5 – Heterogeneous CPU/FPGA architecture for real-time simulation

2.4.2 The execution model

For the execution model, each avionic model can be designed with different versions (i.e. software, hardware, etc.). A common high level model is developed in order to encompass different functions which correspond to different implementations. The necessary data (input, output, current context) is contained in a global data structure (stored in the shared memory in figure 2.5) allowing easier context switch from a software node to a hardware node and vice-versa at runtime and without a full simulation restart.

Figure 2.6 shows how a software/hardware (or vice-versa) context switch can happen at runtime. In fact, the HwFunction() and SwFunction() share the same data context and the I/O data structure in order to perform the calculation node switching more efficiently. Let us highlight that the HwFunction() communicates with the hardware core using the Xillybus [14] solution (will be detailed in the next subsection), and thus bringing a total transparency for the system. Our solution avoids additional timing cost for the software-hardware context switch. As a reconfiguration scenario, an anticipated overload alert can be generated for an avionic model re-allocation in order to avoid the violation of timing constraints and thus the failure of the simulation phase. In our environment, the decision of the initial mapping is taken by an exact method [9] [10] [35] while runtime allocation is ensured by a heuristic [36] depending on the simulation scenario requirements. Our runtime mapping heuristic is developed to deal with the model overloads and to make a decision about the dynamic context switch according to the available software or hardware implementations [2].

14. http ://xillybus.com/

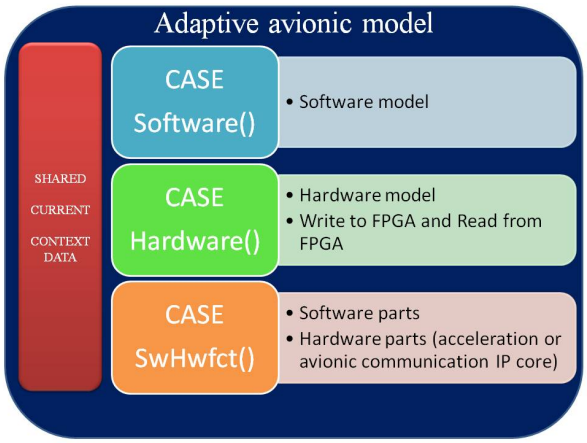

FIGURE 2.6 – Adaptive avionic model

2.4.3 Xillybus : making FPGAs talk PCIe easier

Due to the PCIe bus complexity, the communication in a heterogeneous architecture remains complex. Most of the time, all PCIe capabilities are not even required (i.e. prototyping), an abstracted communication level would improve the design cycle. Xillybus proposes a simple interface for the FPGA and the application designer : The FPGA application logic connects to the IP core through standard FIFOs (for read and write), and the user application on the host machine (Microsoft Windows or Linux) performs plain file I/O operations. Streaming data move naturally between the FIFO and the file handler opened by the host application. There is no specific and intrusive API involved, allowing the hardware and software designers to focus on the requirements of their application. This setting relieves the FPGA designer completely from managing the data traffic with the host. Rather, the Xillybus core checks the FIFOs ”empty” or ”full” signals (depending on data direction), and initiates data transfers when the FIFO is ready for it. As the number of streams and their attributes are configurable, this solution scales easily as the design requirements expand. Figure 2.5 depicts a simplified block diagram showing the connection of one data stream in each direction. The application on the computer interacts with device files that behave like named pipes. The Xillybus IP core and driver on the host offer efficient data streaming (using DMA) between the FIFOs in the FPGAs and their respective device files on the host. The Xillybus IP core implements the data flow utilizing PCIe transport layer level, generating and receiving Transaction Layer Packets (TLPs). For the lower layers, it relies on Xilinx official PCIe core, which is part of the development tools. Making the communication simple is sometimes not enough, the goal is to find the best trade-off between simplicity, reliability, design time and performance in order to address all requirements of our application [6].

Mapping and scheduling of tasks on CPU/FPGA system :

The usage of CPU/FPGA architecture in the context of simulation and test environment needs tools to map efficiently tasks on the heterogeneous computing nodes. As all the connections between the different nodes are allowed, the communication delays are also heterogeneous. Targeting the initial mapping, we focused on the mathematical modelling of a scheduling problem in a heterogeneous CPU/FPGA architecture with heterogeneous com-

munication delays in order to minimize the makespan, C_{max}. This study was motivated by the quality of the available solvers for Mixed Integer Program (MIP). The proposed model includes the Communication delays constraints in a heterogeneous case, depending on both tasks and computing units. These constraints are linearised without adding any extra variables and the obtained linear model is reduced to speed-up the solving with CPLEX up to 60 times. The computational results show that the proposed model is promising. For an average sized problem up to 50 tasks and 5 computing units the solving time under CPLEX is about few seconds which is a reasonable time for the initial mapping of the system [9] [11]. We highlighted that the particular case of homogeneous multiprocessor scheduling with heterogeneous communication delays has been already resolved in [10]. Actually, we proposed a new MIP formulation that drastically reduces both the number of variables and the number of constraints, when compared to the best mathematical programming formulations from the literature [10].

Our investigation concerned also the development and the comparison of efficient heuristics that focus on the dynamic mapping of new applications at run-time, and the dynamic reconfiguration to avoid the real-time constraint violation. The Greedy heuristic LPT-Rule (Longest processing Time Rule), HEFT (Heterogeneous Earliest-Finish Time) Heuristics are explored in our work. Compared to an exact methods, these heuristics offer a good optimality in a time magnitude of milliseconds [36] [35].

2.4.4 Real-time simulator supporting heterogeneous CPU/FPGA architecture

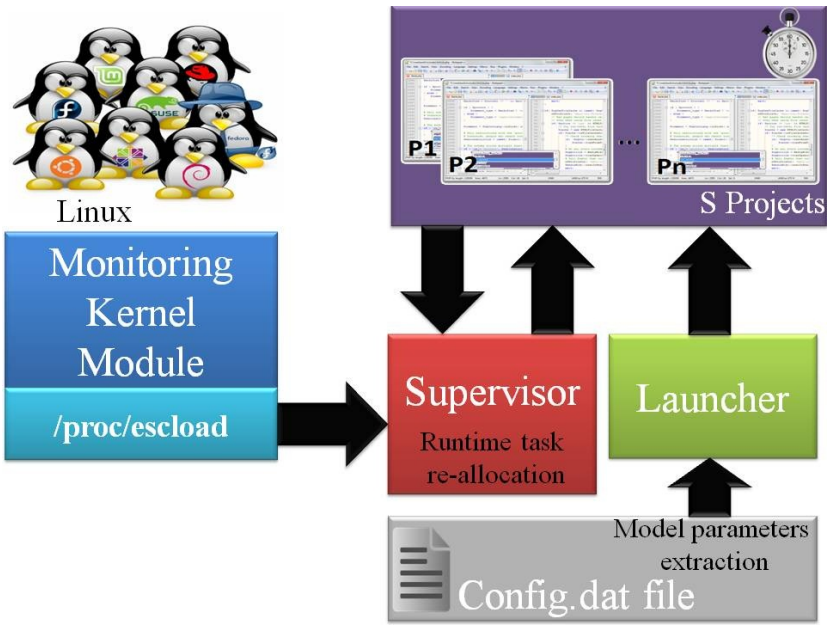

FIGURE 2.7 – The CPU/FPGA simulation environment

In this work, we consider application domain related with soft real-time requirements where response time is in the order of milliseconds. Figure 2.7 presents our modular simulation environment composed mainly of the simulation project, the launcher, the monitor, and the supervisor. This software solution ensures many functionalities : monitoring of compu-

tation nodes, task launch and mapping on different types of architecture including heterogeneous architectures CPU/FPGA in order to respect application real-time requirements, supervision and runtime reconfiguration. Speed and genericity are required in our simulator in order to fulfill real-time constraints.

A simulation project is a set of synchronized models respecting the precedence graph. In order to take profit from the speed of the hardware available resources, each simulation model can have different implementations as shown in figure 2.6 :

- SW, exclusively software (soft) ;
- SWXHW, exclusively software and exclusively hardware (soft $\oplus$ hard) ;
- HW, exclusively hardware ;
- SWSHW, partially software and hardware (soft & hard split), a part of the code runs on FPGA and the other runs on CPU.

A model is a piece of code to be executed on a core and/or on a FPGA. Every model has many specifications : input and output data, version, required execution time, etc. Each task contains a specific model, called “source model” which embeds a timer that periodically wakes the task. This timer is also used as a watchdog to stop any model of the task at the end of the period in case of time-overflow. Each model periodically computes its load and asks for migration or signals an overflow to the supervisor if needed. Each model is able to migrate “by itself” from one core to another one or from one core to the FPGA if possible. The simplified basic principle of functioning of a model is the following :

1. If not the “source model”, wait for all input data, otherwise wait for timer-tick ;
2. Run the simulation code ;
3. Send all output data to other connected models (successors) ;
4. Go to (1).

To start a simulation project, the launcher proceeds in several steps :

1. Collect required data from a configuration file. This file provides information about the simulation project such as the number of tasks, the number of models of each task, the nature of each model (SW, SWXHW, HW or SWSHW) the initial optimal mapping of the models on the CPUs/FPGA (i.e. the affectations) resulting from a mathematical model. All information are stored in an internal data structure in memory.
2. Create communication and synchronization channels between the different models.
3. Create communication and synchronization channels between the models and the supervisor.
4. Connect communication and synchronization channels between the launcher and the supervisor.
5. Send the configuration file-name as a *request* to the supervisor (which acts as a simulation server) which equally read the configuration file in a data structure.
6. Wait for the answer of the supervisor before continuing.
7. Create and allocate every model to its corresponding core (or FPGA) mentioned in the configuration data structure. All models are in a wait-state, excepted the “source model".
8. Wait for the end of (at least) one model. Ensure that all models are stopped at the end of the simulation and destroy all channels.

The monitor is a standalone module in the Operating System Kernel. Consequently it can easily and quickly access the OS Kernel data with an insignificant overhead on the operating system. It calculates the load of the different CPU cores every millisecond. In fact, it mimics Linux standard commands like "`top`" and "`htop`" with a much higher sampling rate. The results are available in the entry `escload` of the pseudo file system `/proc`. The supervisor can access to this entry anytime. In fact, the supervisor, the main component of our architecture, needs the different cores load and each model execution rate in order to ensure new mapping if necessary. It can be considered as a simulation server. When no simulation is running, it waits for a request from the launcher. As soon as the launcher sends to it a new configuration file-name, the supervisor proceeds in several steps :

1. Read data from the configuration file and store information in an internal data structure in memory. The data structure provides many information, including models (initial) affectation.
2. Check if communication channels between the launcher and the supervisor are correctly established and give its approval for a new simulation to the launcher if this is the case (reject the current simulation project otherwise).
3. Connect communication and synchronization channels between the models and the supervisor.
4. Wait for a simulation-alert.

In step 4, the supervisor spends its time waiting for simulation-alerts (i.e. events or requests). Four kinds of alerts might occur : request for migration, too many consecutive migrations alert, overflow alert, and end of simulation. The request for migration is sent by a model as soon as its execution load is greater than a predefined threshold, set to 85% of the model execution period in the presented example. The overflow alert is sent by a model as soon as its execution load is greater than a second threshold, set to 92% of the model execution period in the presented example.

When a request for migration occurs from a model, the supervisor retrieves the current model load and asks for CPU-loads to the monitor through the entry `escload` of the pseudo file system `/proc`. Then it starts the mapping heuristic in order to reallocate the model to a core or to the FPGA if this is possible. The new allocation is sent to the model which is able to migrate "by itself" from one core to another one or from one core to the FPGA if possible. The supervisor might also decide to stop the current simulation if no solution can be found by the heuristic, i.e. no core with sufficiently low load and this model cannot be run on the FPGA. When an overflow or too many consecutive migrations alerts occur then the supervisor decides to stop the simulation. At the end of a simulation, the supervisor computes several statistics, generates a new configuration file containing the last mapping of simulation models and additional data, sends a request to the launcher (in order to stop all models) and starts waiting for a new simulation. More details about our real-time simulation environment can be found in [13].

2.4.5 Design methodology

The above described architecture is attractive for heterogeneous system prototyping and performance evaluation, however we need tools to help software designers to map application on such system. In current industrial practice, manual coding is still widely adopted

in the development of heterogeneous architectures, which is clearly not suited to manage the complexity intrinsic in these systems. For designers, this approach is very tedious, error-prone and expensive. To overcome this challenge, we present a design methodology that covers the different development steps from software specification to the system implementation as shown in figure 2.8. First, we are considering a software application presented as a task graph containing different communicating models (M0, M1, etc.). All applications are not adequate to be implemented onto heterogeneous CPU/FPGA architectures ; a complete analysis of the source code is needed to verify if a hardware implementation could bring better performances. In order to leverage the parallelism of the multi-core CPU/FPGA architecture, tools such as Vector Fabrics Pareon [139] and GAUT [77] can find all data dependencies by analyzing the C or C++ source code and extract the parallelism intrinsic in the application. Pareon analyses partitions and maps applications on specific platforms as heterogeneous ones. It can also estimate the performance of the parallelized software before implementing it. Moreover, it can trim any overhead in your hardware to reduce cost and ensure that all critical behaviours in your program are exercised. After this analysis, the developer will have key information for source code optimization. To perform the application mapping, system resources and application constraints are needed. This step requires a specific heuristic method to resolve a multi-objective exploration problem. After the mapping step, we need to develop some user hardware applications from the existing models (M1, M2, etc.). To make this step more efficient, tools such as Riverside Optimizing Compiler for Configurable Computing (ROCCC) [140] can focus on FPGA-based code acceleration from a subset of the C language. ROCCC does not focus on the generation of arbitrary hardware circuits. Its objectives are to maximize parallelism within the constraints of the target device, optimize clock cycle time by efficient pipelining, and minimize the area utilized. It uses extensive and unique loop analysis techniques to increase the reuse of data fetched from off-chip memory. The communication synthesis step consists of generating the required CPU-to-CPU or CPU-to-FPGA communication interfaces depending on the selected mapping. Having all source code for a CPU/FPGA implementation, the compilation step, using GCC and ISE from Xilinx can be easily performed in order to map all the application onto the system [5].

2.5 Reconfigurable computing for test

As stated in Section 2.3.1, the test avionic domain calls for an additional hardware in order to communicate with avionic equipments. In current industrial practices, one of the biggest challenges of relying on different PCBs (Printed Circuit Boards) for different requirements, is the hardware obsolescence issue. Ever-changing application requirements demand the customization of the I/O bus interfaces. Changing the hardware means redesigning the entire board, with a lot of Non-Recurring Engineering (NRE) cost and significant time-to-market. The VITA (VMEbus International Trade Association group) FMC (FPGA Mezzanine Card) standard [142] solves the I/O obsolescence issue partially, with a single 400-pin connector with a potential overall bandwidth of 40 Gb/s. This essentially means that the I/O bus interface of a PCB is designed separately as a module and interfaced with the board using the FMC connector. Thus, every time an I/O bus interface needs a change, just the module changes, thereby avoiding a complete redesign. For the test phase, the FPGAs can be used for more than just computational purpose in order to improve the system perfor-

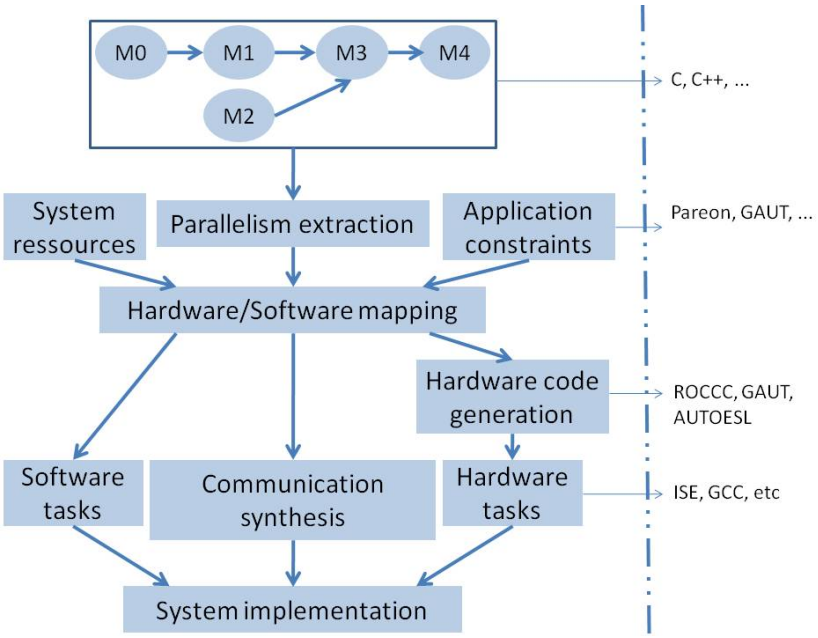

FIGURE 2.8 – Application design methodology for heterogeneous CPU/FPGA system

mance. The introduction of FMC I/O standard has given a new purpose for FPGAs to be used as a communication platform. Taking into account the features offered by FPGAs and FMCs, such as flexibility and modularity, we have redefined the role of these devices to be used as a generic communication and computation-centric platform. Thus, in addition to the avionic models, FPGAs will implement I/O IPs such as ARINC429 in order to perform the communication with the UUT.

A new modular, runtime reconfigurable, Intellectual Property (IP)-IP communication-centric hardware is proposed for avionic test application domain as illustrated in figure 2.9. The hardware architecture is composed of standard machines running virtual avionic models coupled with FPGA boards equipped with FMC connectors. These connectors ensure through the I/O interface just the physical connection with the avionic equipment. The communication protocol is implemented as an IP hosted on the FPGA and data are transmitted via the FMC. Thus, the test phase for a given equipment requires the instantiation of the appropriate I/O IP protocol while the other avionic models remain virtual. Some models can be also hosted on the FPGA as the same level of the I/O IPs which can reduce significantly the communication delays. We can rely on several FPGA boards in order to consider several avionic UUT on the loop.

With such architecture, avionic IP cores can be explored by system designers in different scenarios depending on the application requirement. Therefore, depending on the test scenario, the user can choose a communication IP core to be configured dynamically. This removes the need to have multiple/redundant systems, each for a different protocol. Moreover, when the IP core is reconfigured, the communication channels with the FMC is also reconfigured dynamically along with the protocol. In case the FMC module does not provide a corresponding interface for the communication core being reconfigured (which can be detected using the I^2C EEPROM in the FMC module), it can be swapped with another appropriate FMC module. Thus eliminating the need to redesign the entire board based on a new I/O

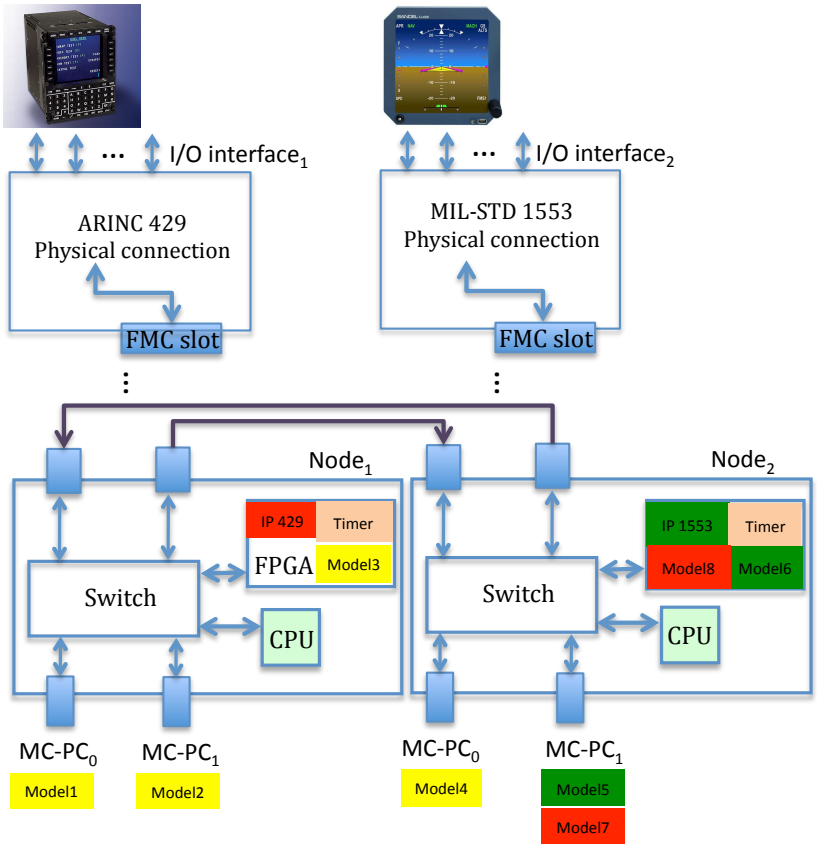

FIGURE 2.9 – Hardware architecture for test system

interface requirement.

In the next sub-sections, we will detail the implementation of three widely used avionic I/O communication protocols : ARINC429, CAN Bus, and the MIL-STD-1553. These protocols are designed in the frame of an IP-based approach for the test phase.

2.5.1 Examples of avionic communication protocols

CAN BUS : The CAN controller designed, implements the Data Link Layer as defined in the document "BOSCH CAN Specification 2.0" [15]. It implements a serial communication which efficiently supports distributed real-time control with a very high level of security. The design has two communication channels. The CAN bus protocol supports up to 16 channels, and has a maximum bandwidth of up to 1 Mbps. The transmission can be programmed to a random frequency using the configuration registers. The architecture of CAN controller is shown in figure 2.10. CAN bus finds its application in several automotive applications.

CAN works in the principle of automatic arbitration-free transmission. A CAN message that is transmitted with highest priority will succeed, and the node transmitting with the lower priority message will sense this wait for transmission. This is achieved by using the notion of dominant bits and recessive bits where dominant is a logical 0 and recessive is a logical 1. Therefore, in a physical implementation of the bus, if one node transmits a domi-

15. http ://esd.cs.ucr.edu/webres/can20.pdf

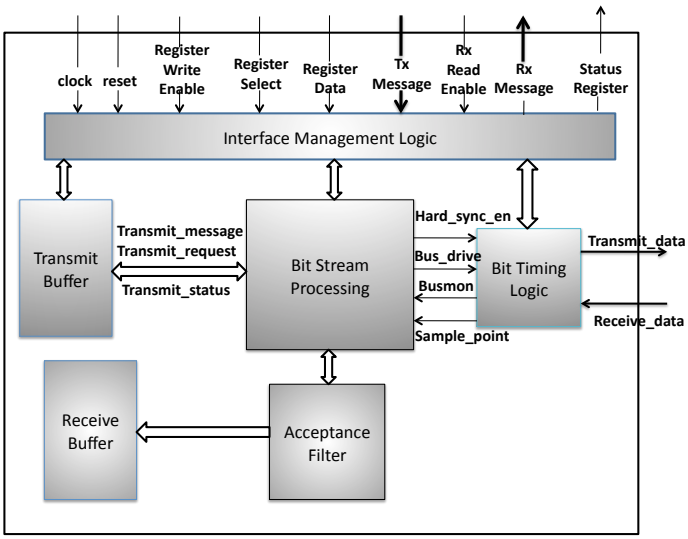

FIGURE 2.10 – CAN bus controller architecture

nant bit and another node transmits a recessive bit then the dominant bit takes priority over the recessive bit. This is implemented as a logical AND between the bits.

Furthermore, when recessive bit is transmitted while a dominant bit is being sent, the dominant bit will be displayed, to indicate a collision. A dominant bit is asserted by creating a voltage across the wires while a recessive bit is not asserted on the bus. Any voltage difference asserted is seen by all the other nodes. Thus there is no delay to the higher priority messages, and the node transmitting the lower priority message automatically attempts to re-transmit six bit clocks after the end of the dominant message.

When a differential bus is implemented, a Carrier Sense Multiple Access (CSMA) scheme is often implemented. If more than two nodes transmit at the same time there is a priority based arbitration scheme. The CAN implements a priority based arbitration when a differential bus is employed. This makes the dominant message delay free, therefore making the CAN bus protocol suitable for real-time communications.

During such arbitration, each transmitting node monitors the sate of the bus, and compares the received bit with the bit that is transmitted. When a dominant bit is received, while a recessive bit is being transmitted, the node stops the transmission. This arbitration is performed during the transmission of the identifier field (ID). Each node that wants to transmit on the bus sends an ID with dominant bit starting from the highest bit. As soon as their ID is a larger number (lower priority) they will be sending 1 (recessive) and see 0 (dominant), so they back off. At the end of the transmission of the identifier field, all the nodes would have backed off, except the one with the highest priority.

The ARINC429 : is an application-specific technical standard for the avionic data bus used on most higher-end commercial and transport aircrafts. It defines electrical characteristics, word structures and protocol necessary to establish an avionic bus communication. For ARINC429, messages are transmitted at a bit rate of either 12.5 or 100 Kbps to other subsystems. The design supports up to 16 Transmit and 16 Receive channels. The architecture of ARINC429 was designed according to the ARINC protocol specifications [16]. The architecture

16. http ://www.aim-online.com/pdf/OVIEW429.PDF

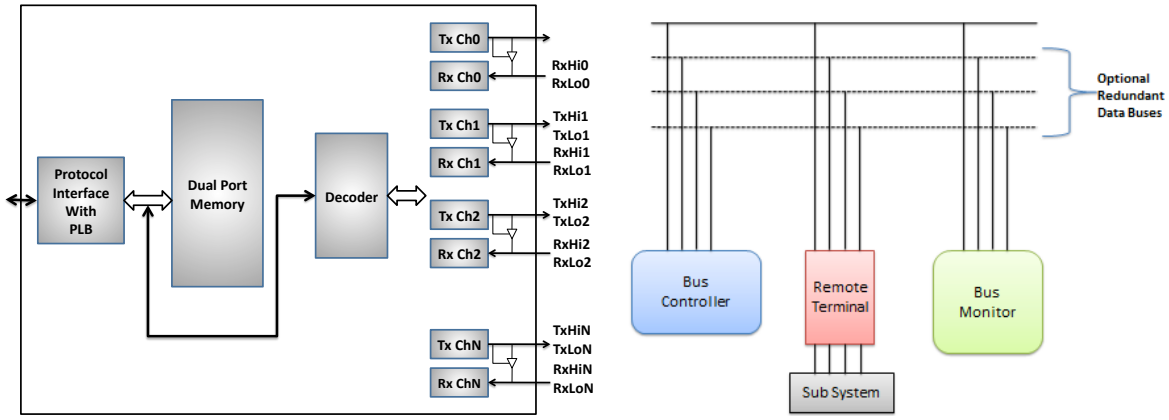

FIGURE 2.11 – The ARINC429 bus architecture (left) and the MIL-STD-1553 bus architecture (right).

of our ARINC429 core is shown in figure 2.11 (left).

ARINC429 standard uses a self-clocking, self-synchronizing data bus protocol where transmit (Tx) and receive (Rx) are on separate ports. Data words are 32 bits in length and most messages consist of a single data word. The physical connection wires are twisted pairs that carry balanced differential signaling. The transmitter transmits either 32-bit data or the NULL words. A single wire pair is limited to one transmitter and up to a maximum of 20 receivers. The protocol allows for self-clocking at the receiver end, thus eliminating the need to transmit clock information. ARINC429 is an alternative to MIL-STD-1553 standard.

MIL-STD-1553 : The MIL-STD-1553 is originally serial military standard protocol that defines the mechanical, electrical, and functional characteristics of a serial data bus. It is now also being used in spacecraft On-Board Data Handling (OBDH) subsystems, both military and civil. The architecture of the bus system consists of a Bus Controller (BC) controlling multiple Remote Terminals (RT) all connected together by a data bus providing a single data path between the bus controller and all the associated remote terminals. The RT is used to interface with other user defined subsystems. There can also be one or more Bus Monitors (BM) ; however, they are not allowed to do any data transfers, and are only used for recording the data for analysis. The protocol also supports several data buses to provide multiple redundant data paths upto a maximum of 4. The protocol follows very strict timing constraints and requirements and provides a maximum bandwidth of 1 Mbps. We have developed our own IP core according to the MIL-STD-1553 specification [17] and used it to evaluate our system. The architecture of the MIL-STD-1553 is given in figure. 2.11 (right).

2.5.2 Towards the convergence between the simulation and the test domains

Using the FPGA as a centric computation component for simulation as well as a communication centric component for test leads to the convergence towards a unified environment for simulation and test. We promote that *all is IP* (avionic models and I/O communication protocols) in our environment. For a given simulation or test project, we have to instantiate the appropriate IPs at the initial phase. In different scenarios, theses IPs can be managed at

17. http ://http ://mil-std-1553.org/

runtime. For instance, a software avionic model can be replaced with a hardware implementation when more performance should be delivered. We can also switch dynamically between a simulation and test phases with just replacing the virtual model with the appropriate I/O protocol to communicate with the UUT using the DPR feature of the FPGA. Considering the number of avionic systems (from 10 to 100) that can be embedded depending on the helicopter range, several computing nodes are necessary. Hence, a unified environment for simulation and test can be a network of heterogeneous CPU/FPGA nodes with different I/O interfaces. A supervisor is needed to manage all the available resources at runtime. As all is considered as IP, a new simulation or test project is assimilated to resource allocation problem. We need also to deal with the communication and the reconfiguration models in order to meet real-time constraint and dynamic re-allocation. These objectives are considered as future works, and we will focus only on one computing node as it will be illustrated in Section 2.7.

2.6 Towards reconfigurable computing for embedded avionic applications

After the validation of the avionic system through the simulation and the test phases, we need to integrate the FPGA-based solution on the aircraft as a standalone hardware gathering the computation and the communication parts. At this level, we meet the conventional methods and industrial tools used for fault tolerance, verification, and certification in order to address special requirements that demand powerful and highly reliable designs. Significant research and industrial efforts have been devoted at the circuit and EDA (Electronic Design Automation) levels to reach this objective. As an example, FPGA device manufacturers are collaborating with EDA tool vendors to resolve difficult problems like providing triple redundancy for dealing with single-event upsets (SEUs) issues in avionic applications.

Today, Xilinx offers on the 7 series FPGAs automatic detect and correct circuitry (CRC/ECC) with Partial Reconfiguration. This technique scans and corrects 2-bit upsets in 20-30 *ms* for most devices and enables SEU logging and tracking. CRC/ECC operates independently of user design. Through the Mentor Graphics [18] (an EDA technology leader) and Xilinx collaboration, the tool Precision Hi-Rel synthesis software is provided. In addition, Xilinx offers TMRTool software for Space and other extreme reliability applications. Other FPGA vendors such as Actel and Altera are also providing their commercial solutions.

Today, Triple-Modular Redundancy (TMR) techniques are widely used to mitigate radiation effects, but TMR requires substantial overheads such as increased area and power requirements. In order to reduce these overheads while still providing sufficient radiation mitigation, authors in [96] propose a Reconfigurable Fault Tolerance (RFT) framework that enables system designers to dynamically adjust a system's level of redundancy and fault mitigation based on the varying radiation.

As the number and the complexity of embedded avionic systems have grown in nowadays aircraft, it became necessary for the FAA (Federal Aviation Administration) to establish a baseline of minimum design flow steps for avionic equipment. DO-254 [19] was formally recognized in 2005 as a standard for ensuring the highest level of safety in electronic avionic systems. It provides guidance for the design assurance of Complex Electronic Hardware

18. http ://www.mentor.com/
19. http ://www.do254.com

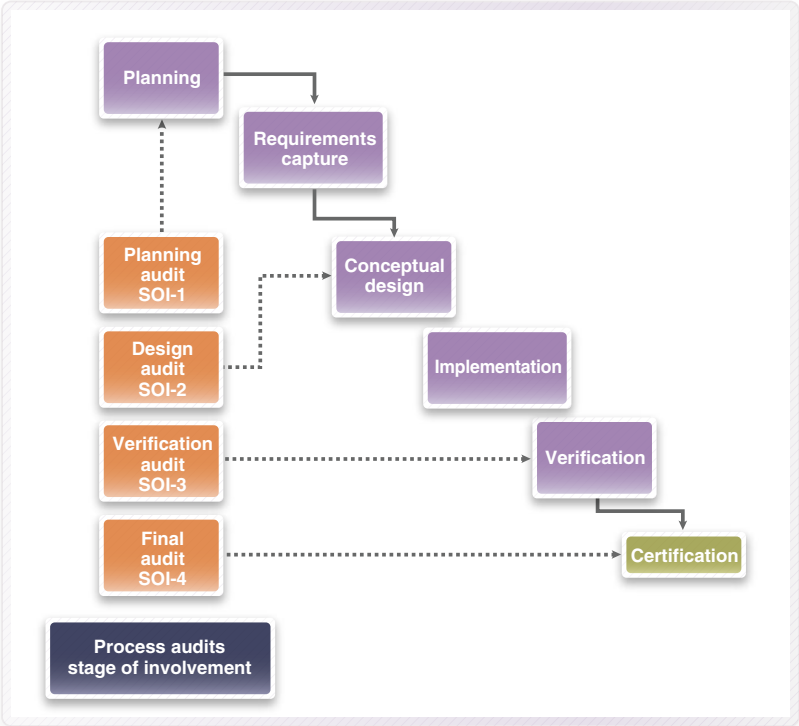

FIGURE 2.12 – DO-254 process flow from Synopsys

(CEH) in airborne systems and equipment for use in aircraft or engines [20]. However compliance with DO-254 requires assistance from the FPGA tools vendors because there are requirements to provide documentation and traceability. The tools vendors such as Synopsys [21] have been doing a lot to provide easier and more comprehensive ways to ensure that compliance. In fact, tool assessment is a part of the DO-254 process that is meant to ensure that the tools used for hardware design and verification perform correctly.

A DO-254 compliant design is specified using a set of formal requirements. As part of the certification process, the applicant must prove that their implementation meets all of these requirements. A graphical illustration of the typical process flow is shown in figure 2.12.

The first step in the DO-254 process flow is the design specification using formal requirements leading to a verification plan that should be tracked along the process. The next step is the design implementation. FPGA implementation is typically verified through RTL simulation, to validate design intent, and code coverage analysis to ensure 100% coverage of all possible input signal combinations across a series of applied tests. However, while simulation results can be easily visualised, analysed, compared and requirements traceability easily maintained, the design behaviour in real hardware cannot be easily traced back to simulation because it is simply not possible to achieve 100% specification coverage once the FPGA is physically mounted onto a circuit board.

To have full traceability you need to be able to compare the behaviour of the physical outputs of the device with their corresponding RTL simulation results. However, rarely we

20. http ://www.atego.com
21. http ://www.synopsys.com/

can drive physically the hardware with all combinations of stimuli. Even for the inputs, the creation of test vectors is an intensive and time-consuming manual task.

Accordingly, performing verification to satisfy DO-254 at the board-level is not only challenging and risky, it is sometimes just not feasible within project time-scales ; which is why engineers are increasingly adopting a so-called in-hardware verification methodology. Aldec [22] provides a Compliance Tool Set that enables DO-254 compliant design and implementation flow.

In the following subsection, we give an example of embedded runtime reconfigurable system. It is necessary to follow the above described steps to reach a certified design, however this objective is not in the scope of this research work.

2.6.1 Example of runtime reconfigurable system

Figure 2.13 illustrates an example of architecture implementing modular runtime reconfigurable system. The setup consists of an embedded processor attached to a few peripheral devices and the avionic communication protocols via the Processor Local Bus (PLB). These I/O protocols communicate with the external sub-systems via the interface provided using a FMC module.

In order to compare and contrast the efficiency of our runtime reconfigurable communication system, with the traditional redundant systems, we have two different types of architectures. First, only one communication IP core is present at a time and can be swapped on-demand. Second, all the IP cores are present at the same time. An example of image encoding (JPEG) application was chosen to evaluate the architecture and our IP cores. The software interface is provided by a Xilinx Microblaze processor with some peripherals attached using a Processor Local Bus (PLB), as shown in figure 2.13. A Xilinx ICAP controller is used to perform partial reconfiguration. We have chosen PLB interface instead of Fast Simplex Link (FSL) because, the IP cores are serial communication protocols and require the configuration of only few registers for their operation. However, on the other hand, JPEG Encoder [114] requires a high-bandwidth communication link with the processor, for processing and data transfers. Therefore, it is connected to the processor using Fast Simplex Link (FSL). However, if need arises the JPEG application can also be implemented as a dynamic part. Compared to a real implementation, the proposed architecture should rely on a certified technology. However, the functionality of the system remains the same. In addition, other design metrics should be involved such as power consumption and area utilization.

Due to the attraction of heterogeneous architecture, today a cutting-edge embedded technology appears gathering multi-core processor and reconfigurable logics on the same chip. The Xilinx Zynq 7000 Extensible Processing Platform (EPP) is an example of such circuit embedding a dual Cortex A9 processor and tens of thousands of programmable gate arrays. The Zynq 7000 [23] combines the software programmability of a processor with the hardware programmability of an FPGA, resulting in unrivaled levels of system performance, flexibility, scalability while providing system benefits in terms of power reduction, lower cost with fast time to market. Undoubtedly, the essential features of systems to combine software and hardware programmability (sequential or parallel) and to reconfigure themselves (at the hardware or the software level) at run-time comes with additional complexity in the different design flow steps.

22. http ://www.aldec.com
23. http ://www.xilinx.com/

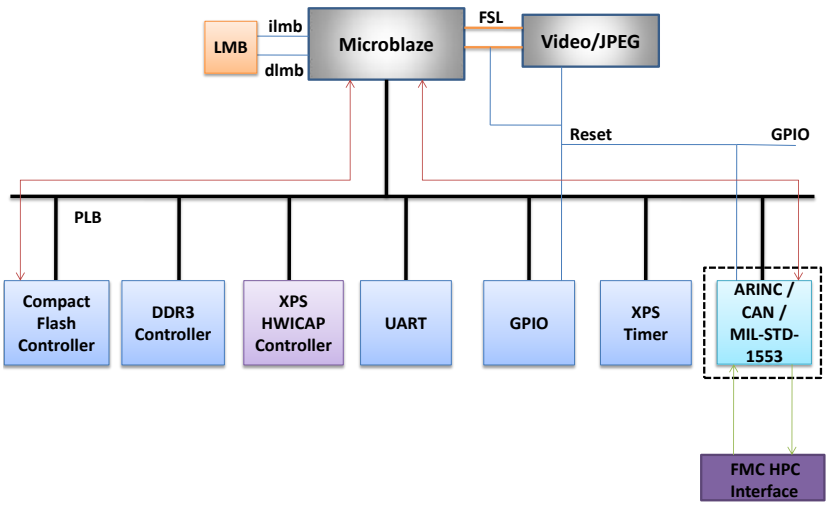

FIGURE 2.13 – Embedded hardware architecture with dynamic reconfigurable system

2.7 Experimental results

In this section, experimental results will cover all the design steps in order to underpin the industrial relevance of the proposed reconfigurable-centric design process for avionic systems.

2.7.1 Simulation environment results

To test our solution of heterogeneous CPU/FPGA hardware for real-time avionic simulation, a hardware experimental environment is essential. The hardware environment is based on a bi-processor Intel Xeon E5520 (quad-core) 2.27GHz, 16-GByte DDR3 memory, and a ML605 Virtex-6 Xilinx board. The FPGA is plugged in the mother board through a PCIe slot that can support 2.5 GBytes/s throughput as illustrated in figure 2.14. Our software environment relies on Linux Debian and has been successfully tested with kernel versions ranging from 3.2.0-amd64 to 3.10.5-amd64. It is released under a proprietary software license. To satisfy soft real-time requirement, frequently imposed in industrial domain, we opted for processor affinity. In order to avoid the timing constraint violation, we have modified the standard kernel configuration in order to reduce the latencies. The CPU Frequency Scaling are disabled to keep the cores at their maximum frequency. We have also disabled the swap capability to be sure that no model will be "swapped". Another element that guarantees real-time requirements is the FPGA. However, its utilization can be useless because of the communication latencies with the CPU. In order to fulfill real-time requirements, it is important to find the best trade-off.

As stated before, the main objectives of using the FPGA at this level is improving performance versus software implementation, achieving high simulation speed-up, and fulfilling real-time requirements. To do so, we will analyse different avionic models in order to obtain different possible implementations on our heterogeneous multi-core CPU/FPGA architecture. These models are used for an avionic simulation project such as the flight mechanic and the navigation models. According to the needed performance and the real-time constraints, the design will be tuned and improved as much as possible in order to be executed more

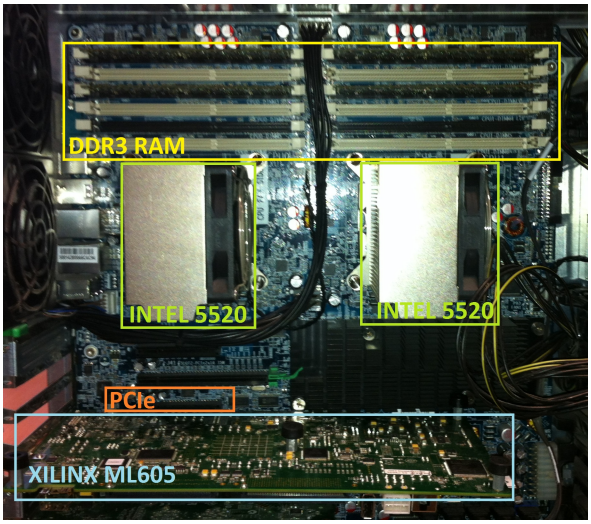

FIGURE 2.14 – Heterogeneous CPU/FPGA for real-time simulation

TABLE 2.1 – Avionic models analysis

Results	Speed-up	Maximum number of useful threads	Synchronization overhead
Model A	1.2	2	1%
Model B	0	1	0%
Model C	2.4	3	0%
Model D	3.5	6	39%
Model E	3	4	29%

efficiently on our architecture considering also switching at runtime between software and hardware configurations or vice-versa.

Table 2.1 summarizes the experimental results obtained by analysing the software models with Pareon tool. First, we measure the speed-up obtained after optimization. Second, threads must be created for parallel implementation strategy. This might be implemented through the use of POSIX calls creating the threads. The maximum useful number of threads is directly linked with the parallelism degree of the application. Indeed, as shown by many parallelism laws, there is always a limit number of useful calculation nodes, it is the same for the maximum of useful threads. But multiple threads means more data synchronisation. That means a less time delay while waiting for data and therefore less latency. Synchronization brings overhead, there is a trade-off between the latency introduced by doing fewer synchronizations with more data and the overhead introduced by doing more synchronizations with less data. Pareon helps the user to get the best trade-off.

Pareon shows a 1.2 speed-up for the model A with low synchronization overhead with only two threads. Model B cannot make profit from a parallelization strategy. For these two models better performances can be achieved just with hardware implementation. Model C, D and E offer higher parallelism degree with low synchronisation overhead for model C.

These models are suitable for multi-core architecture or hybrid multi-core CPU/FPGA implementation by splitting the models into different functions due to the low synchronization overhead.

Using our previous results, we decide to implement model A which is the *Flight Mechanic model* in order to observe the behaviour of such model in a VHDL hardware implementation. With the software version, we obtained a 20μs of execution time with our host.

For the hardware implementation, we target the Virtex-6 Xilinx board (ML605) as execution support. A VHDL implementation of the *Flight Mechanic model* offers a 2μs of execution time with 8% space occupation, this is mainly due to the usage of floating point calculation in this model. This result offers the opportunity to move model A from a processor to the FPGA in the case of timing constraint violation or an anticipated overload.

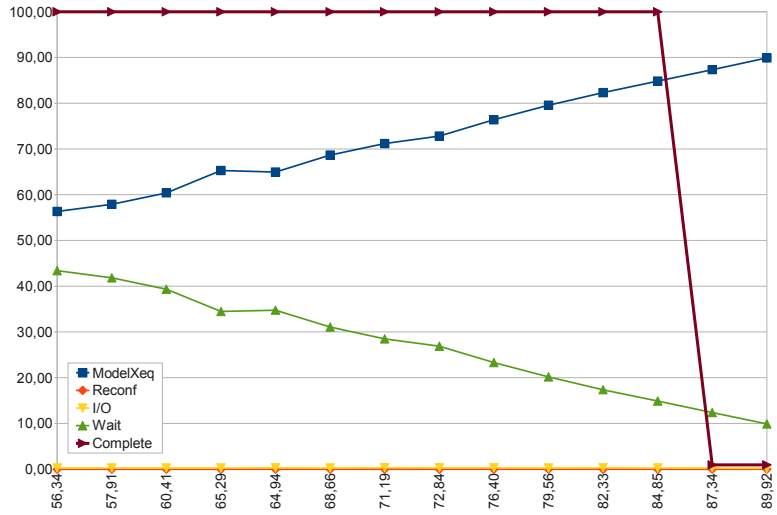

FIGURE 2.15 – Time repartition of a multi-model soft/hard simulation project

In the next experiment, we will consider the simulation loop presented in figure 2.2, the real-time period is set to 10 milliseconds. The *Flight Mechanic model* model is implemented in hardware and hosted on the FPGA while the other models are executed on processor. Our objective is to verify the stability of the simulator according to the global system load. While the load increases and up to a certain limit, many runtime reconfigurations occur without any interruption of the simulation caused by a real-time period overflow. We highlight that the execution time of a software model increases according to the processor load which is not the case of a hardware model hosted on the FPGA.

Figure 2.15 describes simulation model time repartition. The execution time of the model is proportional to the processor's load and inversely proportional to the "wait" duration. "Complete" corresponds to the percentage of successful simulation. Figure 2.15 shows that the I/O time and the reconfiguration time are negligible. As soon as the load exceeds the migration threshold (87%), migrations occur. Then, as the load continues to increase the "too many consecutive migrations alert" occurs and stops the simulation which is mainly due to the software part. This result demonstrates that a heterogeneous CPU/FPGA architecture can be an efficient execution support for real-time simulation in avionic domain without referring to a dedicated and expensive solutions. As a conclusion, we highlight that the FPGA can bring performance in such domain playing mainly the role of a computing hardware

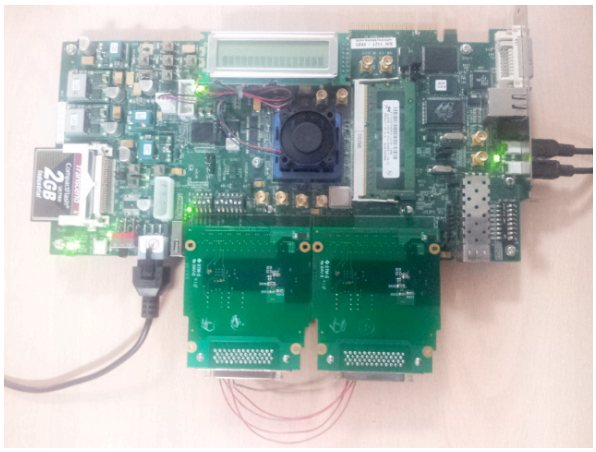

FIGURE 2.16 – ML605 kit showing FMC connector loopback using MIL-STD-1553

accelerator.

2.7.2 Test environment results

For the test environment, we will rely on the same technology in order to establish the communication with the UUT (avionic equipment) according to the generic architecture presented in figure 2.9. In fact, the ML605 board that we are using provides two FMC slots ; one with a High Pin Count (HPC) and the other with a Low Pin Count (LPC). Hence, these slots can be used to host simultaneously two different FMC cards presenting I/O avionic interfaces as shown in figure 2.16. Different communication avionic protocols (ARINC429/CAN Bus/MIL-STD-1553) are implemented and tested in real scenarios. To do so, a Microblaze processor is used in order to configure the registers and initiate data transfers. A number of frames is transmitted to an avionic sub-system using an appropriate communication protocol (ARINC429/CAN Bus/MIL-STD-1553) via a FMC interface. The communication protocol is selected and configured during runtime according to the request. Ideally, the data have to be transmitted to external sub-systems. However, for testing purposes, we have done an external FMC loop-back to verify if the transmission is correct as shown in figure 2.16. We then analyse the FPGA resource utilization, transmission characteristics, I/O pin requirement and scalability for each core. The results are elaborated in the following subsections.

2.7.2.1 FPGA resource utilization

Table 2.2 summarizes the area utilization of each avionic protocol with different configurations. While considering only one IP core active at a time (i.e. a design with only an ARINC429 16 channels or a CAN Bus 16 channels), the consumed area is about 37% of the logic blocks in the FPGA (with respect to Flip Flop utilization). However, When 3 IP cores are implemented at the same time (in this scenario ARINC429 16 channels, CAN Bus 16 channels, and MIL-STD-1553), it consumes over 70% of the resources on the Virtex6 FPGA which is about 50% excess comparing to one IP core active at a time. This is a waste of hardware resources considering the fact that the FPGA can also host avionic models as discussed in Section 2.5. Using Partial Reconfiguration (PR) is very relevant because it clearly elimi-

TABLE 2.2 – Area utilization of the avionic IP cores

	Slices	FFs	LUTs	BRAMs
ARINC429-2 channel	1,912	3,198	7,644	2
ARINC429-16 channel	47,920	44,267	59,400	2
CAN Bus-2 channel	689	1,016	2,754	0
CAN Bus-16 channel	10,176	15,437	41,096	0
MIL-STD-1553- Dual Redundant	1,232	3,432	8,453	16

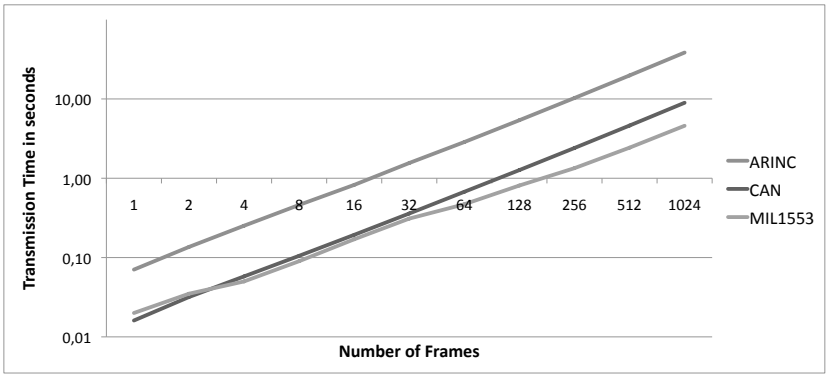

FIGURE 2.17 – Transmission time vs Number of frames

nates the need for having multiple systems (corresponding to different UUTs) for the lack of hardware resources in traditional avionic systems.

2.7.2.2 Transmission times

Figure2.17 shows the performance of our IP cores in terms of time taken for transmitting different number of frames. The system performance has been evaluated up to 1000 frames. The time measurements were done in the software using a Microblaze timer. The transmission and reception is done synchronized according to the protocols' internal clocks with pre-programmed frequencies. However, the time measured, also takes into consideration the overhead of configuring the registers, the overhead caused due to the transfer of status words and the idle time between each transaction. From the graph shown in figure 2.17, it is seen that MIL-STD-1553 is the fastest in transmitting the frames. Although MIL-STD-1553 and CAN buses have the same maximum bandwidth, the fact that MIL-STD-1553 is able to pack more data into a single message and to operate with minimal status feedback, gives it an extra edge in transmitting efficiently.

2.7.2.3 Number of I/O pins and channels

Each protocol requires a specific number of FPGA I/O pins to communicate to the external sub-systems via FMC. The number of pins required is determined by the number of communication channels in the design. Table 2.3 shows the number of FPGA I/O pins required for each protocol according to the number of I/O channels. However, some data in the table are not shown because, CAN bus standard does not support more than 16 channels and MIL-STD-1553 is never used beyond 4 redundant buses. As seen in figure 2.18, the

TABLE 2.3 – FPGA I/O pins requirement for each protocol

Channel count	ARINC429	CAN Bus	MIL-STD-1553
Pins for 1 channel	8	3	6
Pins for 2 channels	16	6	12
Pins for 4 channels	32	12	24
Pins for 8 channels	64	24	-
Pins for 16 channels	128	48	-
Pins for 20 channels	160	-	-

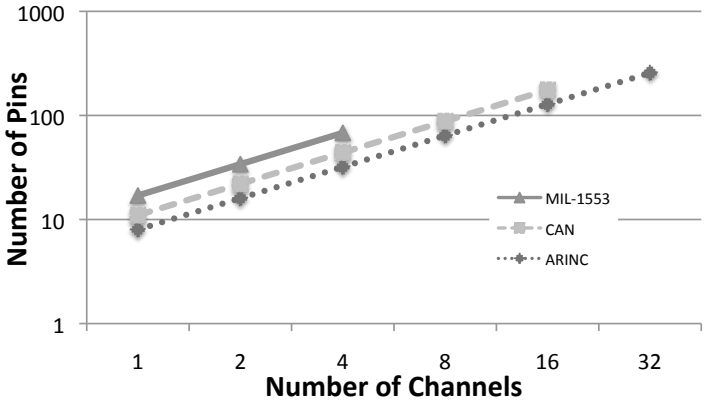

FIGURE 2.18 – Number of pins versus number of I/O channels

number of pins linearly increases with respect to the increase in the number of channels. This parameter is important for the following reasons. It is quite clear that packing all protocols at the same time requires about 230 user I/O pins on the FPGA. However, the number of FPGA user I/O pins that can be allocated for these communication protocols is restricted by the number of FPGA ports and FMC slots. While a mid-size FPGA (as used by many avionic systems) may have anywhere around 300 to 500 pins, using all these pins for FMC I/O will not leave sufficient pins for the other FPGA peripheral devices (i.e. Ethernet, SFP, LCD, memory, back planes, front panels, etc.). On the other hand, a high-end FPGA which may have up to 1200 user I/O pins are usually chosen to pack dense computational logic, hence using them for multiple communication protocols wouldn't be very cost effective.

2.7.2.4 Scalability

The scalability of the system is given in term of number of transmission channels each core can accommodate, which is derived from the protocol specification. ARINC429 is capable of scaling up to a maximum of 20 channels and CAN up to a maximum of 16 channels. However, MIL-STD-1553 is only dual redundant. It is quite important to note that, scalability of a protocol does not increase the bandwidth of the system since the bandwidth of the bus is dictated by the standard itself. However, the number of channels only enables to the system to simply communicate with more number of sub-systems at the same time which is the case when different UUTs are using the same avionic protocol.

TABLE 2.4 – Area utilization of the FPGA design blocks

	Slices	FFs	LUTs	BRAMs	DSP
Microblaze	573	1,737	1,474	101	3
JPEG Encoder	2,482	4,112	6,375	79	10
Peripherals	4,937	4,543	4,051	3	0
Design with 3 IPs + JPEG	79,047 (27%)	68,569 (68%)	127,558 (77%)	150 (36%)	13 (2%)

2.7.3 Embedded avionic application results

The use-case scenario for our architecture is a part of an Unmanned Ariel Vehicle (UAV) avionic system. The task of the system is to travel between different terrains ; to take pictures, to encode sand either store them internally, or to transmit them to a remote system. Depending on the scenario, an appropriate communication protocol (ARINC429/CAN Bus/MIL-STD-1553) has to be selected and configured during runtime. For instance, when a secure transmission is needed, the MIL-STD-1553 is chosen. Our avionic system is implemented respecting the architecture of figure 2.13. The Xilinx EDK and ISE tools were used to generate the bitstream. Initially the partial bitstreams are stored in the Compact Flash memory and are read when requested by the application. The operational frequency of the processor, buses and the peripherals is 100 MHz. A C program is used to initialize and to interact with Microblaze and thus the underlying hardware in order to configure the registers and initiate data transfers. A communication protocol (ARINC249/CAN/MIL-STD-1553) runs in parallel with a JPEG encoder [114]. The captured image is encoded, and is transmitted to an avionic sub-system using an appropriate communication protocol selected during runtime via a FMC interface.

The area utilization of each hardware component is shown in Table 2.4. The Microblaze core occupies 573 slices and 101 block RAMs. This is mostly due to the large embedded memory used to store the software executable (application and operating system kernel). The JEPG encoder needs over 2400 slices, 79 BRAM blocks and 10 DSP blocks, since it buffers the input image frames and performs DSP algorithms to encode the image. About 4900 FPGA slices and 3 BRAMs are occupied by the peripheral devices and the PLB bus interface. These results show the hardware extra-cost of using a softcore processor (the Microblaze) for data transfer and dynamic reconfiguration management.

2.7.3.1 Reconfiguration latency and application profile

We measure the time taken to dynamically reconfigure the system with our communication protocols. All the timing measurements shown are measured from the software. Partial bitstreams are stored in the Compact Flash and read when requested. Figure 2.19 shows the reconfiguration latency versus the bitstream size. From the graph, it is quite obvious that the bigger the size of the bitstream the longer the reconfiguration latency. Bitstreams of size less than 500 KBps require less than a second to be reconfigured. Configuration stream of size 645 KBps which is the size of our bitstream (the IP cores with 2 channels), requires roughly about 1.3 seconds while bitstreams larger than a 2 MBps (corresponds to the IP core with 16 channels) have reconfiguration latency of few seconds. The read queue in the XPS_HWICAP controller buffers the configuration data before it is fed to the ICAP. However, we see that

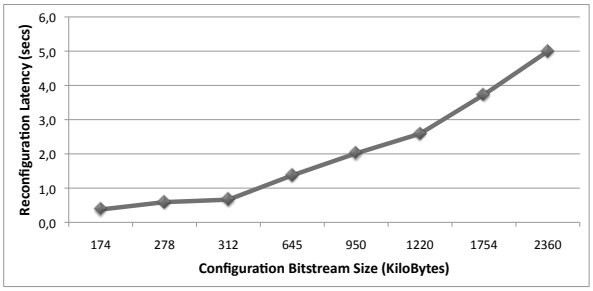

FIGURE 2.19 – Reconfiguration latency versus configuration bitstream size

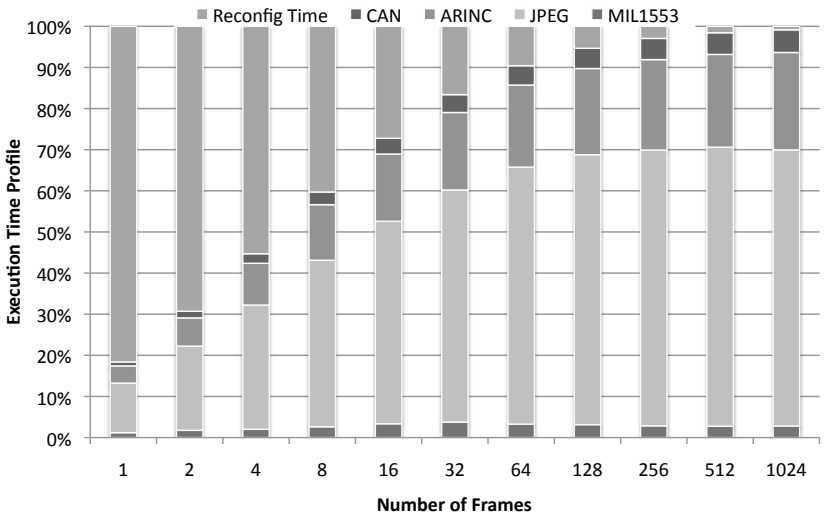

FIGURE 2.20 – Application profile

the throughput of the ICAP controller is less than the theoretical maximum because of the disk access overhead caused by the Compact Flash. It is interesting to note that, the entire avionic protocol is swapped within seconds with uninterrupted system operation.

The reconfiguration latency can be completely hidden in many scenarios as shown in the profile of the application in figure 2.20. From this illustration, it is quite obvious that transmission time is quite negligible compared to execution time of the application (JPEG). It is also seen that the reconfiguration time become also negligible with significant processed data. As Partial Reconfiguration means the ability to dynamically modify the blocks of logic by downloading partial bitstreams while the remaining logic continues to operate without interruption [24], the reconfiguration phase can be anticipated during the application processing.

2.7.3.2 Power estimation

The total power consumption of the design depends on several attributes of the overall design. But it is safe to assume that the bigger the design is the more the power consumption will be, unless special power saving mechanisms such as clock gating is applied, among

24. Xilinx : http ://www.xilinx.com/tools/partial-reconfiguration.htm

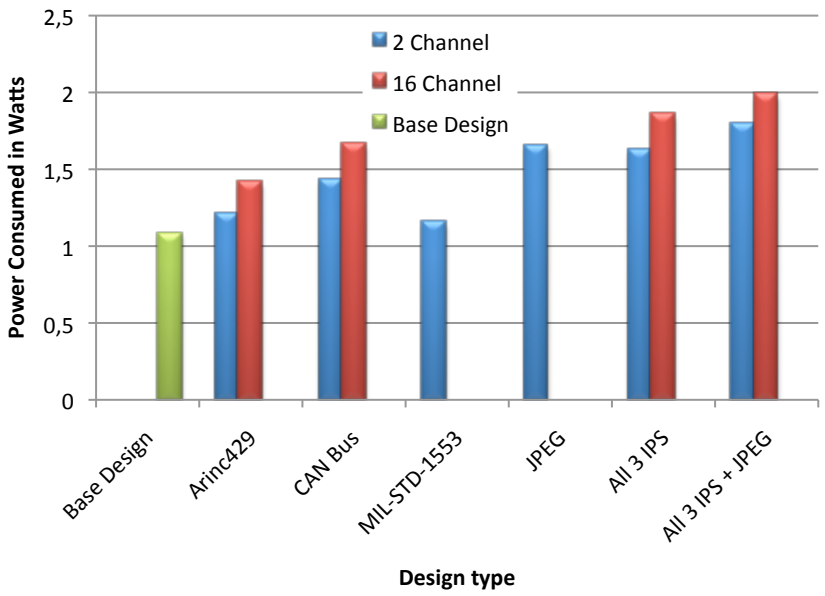

FIGURE 2.21 – Total power consumption of each design

other factors. In the figure 2.21, we are comparing the overall power consumption of each design with respect to the design with ARINC429 16 channels, CAN Bus 16 channels, MIL-STD-1553 and JPEG application at the same time. The maximum power savings are about 400 mW when only one IP is present at a time. Although this difference may not be huge, as mentioned earlier, avionic protocols such as AFDX are far more complex and power-hungry with several Gigabit transceivers operating at the same time.

2.8 Discussion

The Summary. In this chapter, we have presented an FPGA-centric design process for avionic systems. We have redefined the role of the FPGA in the different design steps namely the simulation, the test, and the integration phases. In the proposed process, a particular attention has been given to the smooth transition between the different steps relying on the same technology which yields to a reduced design cost and time-to-market. The main criteria of reconfigurable circuits in terms of performance, flexibility and dynamicity have been exploited to define versatile avionic systems respecting several design constraints (real-time, area utilization, etc.). For the simulation part, first we emphasized the benefits of using emerging CPU/FPGA heterogeneous architecture for high performance and real-time requirements. Second, we brought dynamicity to the heterogeneous system with the definition of an appropriate execution model to support run-time adaptive mapping and we offered the software environment to support such feature. This research work also addressed the challenge of convergence between the simulation and test domains by proposing an FMC (FPGA Mezzanine Card) standard-based communication system. The pertinence of experimental results presents a concept proof of the proposed design process.

Limitations. More investigation on software/hardware dynamic context switching should be performed in order to support preemptible task execution which can lead to a bet-

ter performance execution model. The usage of virtualization techniques, such as exposed in [117] for the management of software and hardware tasks at the operating system running on a commercial heterogeneous platform, allows the reduction of development complexity. High level programming tools are also needed to increase the productivity of designers on dynamic and heterogeneous architecture. As we are targeting a safety-critical domain, the proposed concepts should be certified to be used in the next generation of avionic systems. We should rely on formal verification tools for proving the correctness of the system behaviour.

The future. The target systems (aircraft, helicopters, etc.) are very complex and they are considered as System-of-System (SoS). We already started studying the scalability of the environment to construct a network of heterogeneous computing nodes where the reconfigurable technology will play an essential role. This led to a new patent registration at the INPI [8] in collaboration with Airbus Group. In the future, we will pursue the research on dynamic execution model considering distributed and heterogeneous systems.

Summary	Supervision :	- **George Afonso**, PhD in computer science, graduated in July 2013, now R&D engineer at Safran Group. - **Abdessamad Ait El Cadi**, now Post-doc researcher at the University of Valenciennes, LAMIH Laboratory (January 2013 - December 2014). - **Zeineb Baklouti**, Master degree (April - December 2012), now first year PhD student at the University of Valenciennes, LAMIH Laboratory.
	Patents :	Australian Patent Office [32], Institut National de la Propriété Industrielle [31] and [8]
	Journals :	Optimisation Letters [10], IEEE Transactions on Industrial Informatics [17]
	Selected conferences :	IEEE ETFA 2010 [3], AHS 2011 [4], ASPLOS 2012 [5], IFAC MIM 2013 [35], SIMUTools 2013 [7], RAW 2013 [2], IEEE IESM 2013 [13] [37]
	Realization :	Eurosim simulator [13]
	Funding :	ANRT (CIFRE), Airbus Group industrial grants, Nord-Pas-de-Clais region funding

Chapter 3

Energy/Power-Aware Design Methodology for MPSoC

This chapter presents my contributions since 2009 in LAMIH laboratory and INRIA DaRT team in the field of low power design for homogeneous and heterogeneous Multi-processor System-on-Chip (MPSoC). Mainly, these works was achieved in the frame of the ANR OpenPeople project [25] in collaboration with colleagues from LAB-STICC [26] (University of Bretagne Occidentale), IRISA [27] (University of Rennes 1), and LEAT [28] (University of Nice et Sophia Antipolis). This work also covers the PhD thesis of Santhosh Kumar Rethinagiri started in December 2009 and defended in March 2013 (co-advised by Jean-Luc Dekeyser).

The remainder of the chapter is organized as follows : in Section 3.1, I introduce the main challenges for low power design in complex embedded systems ; Section 3.2 gives a summary of the related works. In Section 3.3, I introduce the OPEN-PEOPLE platform while the necessary background for power consumption is exposed in Section 3.4. Section 3.5 details the first part of my contributions about the power-aware design methodology. In Section 3.6, I present our work devoted to the power modeling. Sections 3.7 and 3.8 illustrate our results about design space exploration and model driven engineering. Finally, in Section 3.9, I discuss the strengths, limitations and future directions to the presented works.

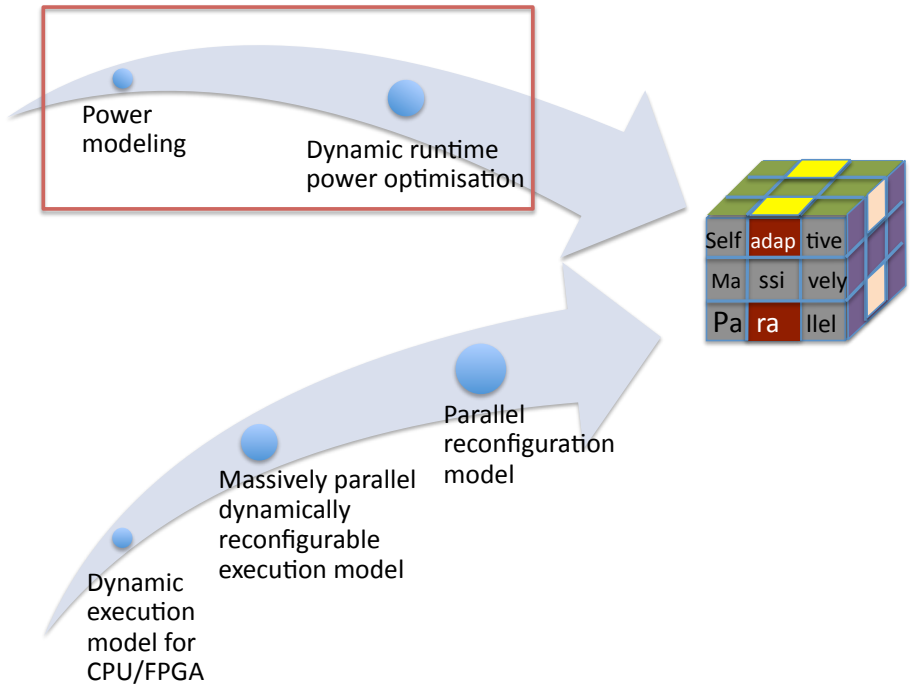

FIGURE 3.1 – Contributions the field of low power design

25. www.open-people.fr
26. www.lab-sticc.fr
27. http ://www.irisa.fr/
28. http ://leat.unice.fr/

3.1 Main challenges for low power design

The increasing complexity of applications and System-on-Chip (SoC) architectures places embedded system designers in front of a very large design space. Exploring this space to reach an efficient solution becomes very difficult, especially when the design must satisfy a large number of constraints, such as power and energy consumption. These constraints have led to introduce the usage of Multi-Processor System-on-Chip (MPSoC) which allow to integrate very complex systems. These MPSoC are generally heterogeneous and can contain different memory structure (Cache, SRAM, FIFO, etc.), processors (GPP, DSP, etc.), interconnecting elements (Bus, Crossbar, NoC, etc.), I/O peripherals, or reconfigurable logic. To use the tremendous hardware resources available in next generation MPSoC efficiently, rapid and accurate Design Space Exploration (DSE) methods are needed to evaluate the different design alternatives. MPSoCs must be designed with custom architectures to balance the implementation constraints between the application needs (i.e. : high computation rates and low power consumption) and the production cost. With the recent sub-micron technologies, power consumption is becoming a critical pre-design metric in complex embedded systems such as MPSoC. Currently, designing low power complex embedded systems is a main challenge for corporations in a large number of electronic domains. There are multiple motivations which lead designers to consider low power design such as increasing circuit lifetime, improving battery longevity, limited battery capacity, and temperature constraints. Nevertheless, the significant increase of complexity in such systems prevents designers from controlling the complete design flow. To guide the designer during the different design choices, the development of an efficient methodology and associated tools for power estimation and optimisation is mandatory. Today, Electronic System Level (ESL) design is considered a vital premise to overcome the increasing design complexity. System designers need an efficient power-aware design methodology and tools to cope with the complexity of MPSoC design. The development of tools for power estimation and optimisation at the system level confronts extremely challenging requirements such as a seamless power-aware design methodology that relies on accurate and fast system power modeling and integrates efficient power optimisation techniques. To be acceptable, the proposed methodology must include all the system-on-chip aspects, i.e. architecture/hardware, application/software, and management/operating system. In the following subsections, we will give an overview of the main challenges related to the low power design of complex embedded systems.

Seamless power-aware design methodology : In current industrial and academic practices, power estimation using low-level CAD tools is still widely adopted, which is clearly not suited to manage the complexity of embedded systems supporting modern applications. In fact, MPSoCs have a huge solution space at the application, the Operating System (OS), and the architectural levels, which makes the DSE complex. This challenge is addressed by several frameworks through the development of ESL tools. The objective is to unify the hardware and software design and to offer a rapid system-level prototyping using virtual platforms. Based on the design step and the requirements like the timing accuracy and the estimation speed, designers could select an appropriate abstraction level to model the software simulating the system. Unfortunately, most of existing tools do not consider the power metric or focus on power estimation for a given abstraction level without overcoming the wall of speed/accuracy trade-off. Multi-level DSE is an unavoidable solution to have a good speed/accuracy trade-off. Indeed, a top-down DSE allows to eliminate fast the undesirable solutions at each design level before reaching physical implementation levels. *In the frame of*

multi-level DSE, designers of embedded applications need a seamless power-aware design methodology that takes into account the power metric at different abstraction levels. In a top-down design methodology, an appropriate power estimation and optimisation tool should be defined according to each abstraction level. The objective is to offer a gradual refinement of the solution space while switching between the design steps.

System power modeling : At the system level, we need power models emulating the behaviour of different parts of the system in terms of consumption. The power modeling process is centred around two correlated aspects : *the power model granularity* and *the main activity characterisation*. The first aspect concerns the granularity of the relevant activities on which the power model relies. It covers a large spectrum that starts from the fine-grain level such as the logic gate switching and stretches out to the coarse-grain level like the hardware component events or the OS services. In general, fine-grain power estimation yields a more correlated model with data and technological parameters, which is tedious for system-level designers. On the other hand, coarse-grain power models are based on micro-architectural activities that cannot be determined easily depending on the complexity of the system. The second aspect of power modeling involves the characterisation of the activities, which requires a huge number of experimental measurements and thus a significant time to extract the power model. The above-described aspects lead to the definition of a power model, which can be represented by a set of analytical functions or a table of consumption values. The selected power model granularity depends on the target abstraction level and the user requirements in terms of estimation accuracy and speed. In the power estimation process, the developed power models interact with virtual platforms in order to grab the strict relevant data (the values of the power model parameters) depending on the design step. *The main challenge is to define a generic power modeling approach that can cover the different abstraction levels and guarantee the coherence of the estimation strategy for a seamless power-aware design methodology*.

The power optimisation techniques : To reduce the power/energy consumption in MPSoC, we can distinguish two main approaches. The first approach is static by the means of design space exploration in order to tune the architecture according to the application requirements. While in the second approach, the power/energy consumption in MPSoC can be reduced dynamically at runtime. At the OS level, two main techniques can be applied : the Dynamic Power Management (DPM) that switches-off the power supply of a part of the circuit and the Dynamic Voltage and Frequency Scaling (DVFS) that tunes the processor clock speed and its corresponding voltage according to the workload (actual or expected) or the battery charge. A lot of techniques have been developed [51] [88] but their efficiency is not evaluated with a sufficient accuracy because, in a majority of cases, they do not rely on realistic time and power models and few of them validate their policy on real-platform due to the corresponding complex and time consuming implementation. *The challenge is to include these optimisation techniques in the design flow as an essential part of the DSE process*.

Software design environment : Solving the challenge of low power design while maintaining acceptable design productivity requires development tools that support abstraction and automation. Therefore, an efficient software approach, such as Model Driven Engineering (MDE) [119], is needed in order to make the SoC design easy and not tedious, by making the low-level technical details transparent to designers. In MDE, models become a mean of productivity. The graphical nature of MDE offered by the Unified Modeling Language (UML) makes the comprehensibility of a system easier and allows users to model their systems at a high abstraction level, reuse, modify and extend their models. Using the automa-

tion offered by MDE, the whole code necessary for the simulation of a SoC can be generated automatically from models describing the system. In order to use the MDE for a high level description of a system in a specific domain such as embedded systems, UML profiles are used. A UML profile is a set of stereotypes that add specific information to a UML model in order to describe a system related to a specific domain. Several UML profiles target embedded systems design such as the Modeling and Analysis of Real-Time and Embedded systems (MARTE) [112] profile. MARTE is a standard profile promoted by the Object Management Group (OMG). *Based on a model driven engineering approach, we need to take the consumption criterion into consideration as early as possible in the software design environment*.

3.2 Related works

Significant research efforts have been devoted to develop tools for power consumption at different abstraction levels in embedded system design. Among the existing tools for low abstraction levels, we can mention SPICE [44], Diesel [118], and PETROL [118] which operate at the RTL level. These tools are fairly accurate, but require significant amount of simulation time. At such low level, tools are used to optimize power consumption of hardware blocks but not to evaluate entirely complex SoC architectures.

To cope with the evaluation time, several tools have been developed for power consumption estimation at the system level. Among the wide-used approaches, we quote tools based on micro-architectural cycle-level simulation such as Wattch [70] and Simplepower [144]. They define fine-grain power models by characterizing component features such as a set of instructions or functional blocks using analytic power laws. The contributions of the internal unit activities are calculated and added together during the execution of the program on the micro-architectural simulator. This approach needs low-level description of the architecture which is often difficult to obtain for off-the-shelf processors. Though using cycle-level simulators has allowed accurate power estimation, the simulation time of complex MPSoC needed to achieve the results is still significant.

In an attempt to reduce simulation time, recent efforts have been done to build up fast simulators using *Transaction Level Modeling* (TLM) [61] [133]. SystemC [113] and its TLM 2.0 kit have become a de facto standard for the system-level description of SoC. The TLM kit proposes different coding styles to offer concepts for loosely and approximately timed models. However, there is no a standard definition for concepts or methodologies that involves power estimation at the TLM level and this aspect is still under research and is not well established. In [109] and [104], a methodology is presented to generate consumption models for peripheral devices at the TLM level. Relevant activities are identified at different levels and granularities. The characterisation phase is however done at the gate level from where the activity and power consumption for the higher level are deduced. Using this approach for recent processors and systems is not realistic. In fact, recent processors have complex architectures ; they may contain several pipeline stages, hierarchical memory system (L1 and L2 cache levels), and specific execution units such as the NEON architecture for the ARM Cortex A8. The power characterisation phase at the gate level of each activity of these blocks needs a huge number of experiments and significant simulation time. Dhawada et al. [80] proposed a power estimation methodology for PowerPC and CoreConnect-based system at the TLM level. Their power modeling methodology is based on a fine-grain activity (processor instruction, data word transmission via the bus, etc.) characterisation at the gate level

which needs a huge amount of development time. Such fine characterisation leads to a high correlation with data, hence authors announced a quite significant power estimation error. Compared to the previous works, our proposed methodology for power estimation also partially uses SystemC/TLM simulation with coarse grain power models.

For the functional level, Tiwari et al. [135] have introduced the concept of Instruction Level Power Analysis (ILPA). They associate a power consumption model with instructions or instruction pairs, which are characterized using measurements on a real chip. The power consumed by a program running on the processor can be estimated using an instruction-set simulator to extract instruction traces, and then adding up the total cost of the instructions. This approach suffers from the high number of experiments required to obtain the model. In addition, it can be applicable only for processors. To overcome this drawback, Laurent [103] et al. proposed the *Functional Level Power Analysis* (FLPA) methodology that was successfully applied on building high-level power models for different hardware components (processor, memory, I/O peripherals, FPGA, etc.). FLPA relies on the identification of a set of functional blocks which influence the power consumption of the target component. The model is represented by a set of analytical functions or a table of consumption values which depend on functional and architectural parameters. Once the model is build, the estimation process consists of extracting the appropriate parameter values from the design, which will be injected into the model to compute the power consumption. Based on this methodology, the tool SoftExplorer [102] was developed. It includes a library of power models for simple to complex processors. Recently, SoftExplorer has been included as a part of Consumption Analysis Toolbox (CAT) [81]. CAT gives relatively precise power estimation results in a surprisingly small time. Indeed, only a static analysis of the code, or a rapid profiling are necessary to determine the input parameters for the power models. However, when a complex hardware or software is involved, some parameters may be difficult to determine with precision. For instance, this is the case of cache miss rates in complex processors. This lack of precision may have a non-negligible impact on the final estimation accuracy, depending on the sensitivity of the parameter. In order to refine the value of sensible parameters in a reasonable delay, we propose in our work to couple SystemC/TLM simulation with functional power modeling. Thus, a reasonable trade-off between estimation speed and accuracy will be reached.

For the reconfigurable circuits (FPGA), several studies have been done during the last years. One of the first modeling proposal has been done in by Garcia et al. in [90, 91]. In this work, the power modeling is measured for the different elements of the circuit (LUT, register, I/O, clock tree, etc.). The power consumption measured in this work concerns the active component, but the configurable memory is not considered, and the reconfiguration aspect is not evaluated. In [89], authors explain how the pipeline of some hardware functions can reduce the power consumption by the reduction of the clock frequency. In general, applying pipeline technique leads to increase the area of the hardware block and hence the static power. One important aspect is then to evaluate the trade-off between dynamic and static power. High-level estimations have also been developed for this type of circuit. In [47], the high level characteristics of the functionality is used to model the power consumption. For example, the frequency of the hardware implementation of a functionality is used to estimate the power/energy consumption. When considering operating system level, the service which ensures the task scheduling and the task placement have an impact on the power consumption, and in particular on the static power consumption. The work presented in [145] shows that the reconfiguration must be done as late as possible to prevent leakage current in the reconfiguration memory, but this can be very interesting if and only if it is

possible to configure a usable area of the circuit in a very low static power. Even if this technique exists, the trade-off between the configuration of this state and the static power saved by this specific configuration must be evaluated. The mentioned above FLPA approach was also applied to develop consumption models for FPGA at the system, algorithmic [84], and architectural levels [85], and to assess the consumption overhead due to hardware reconfiguration phases [127].

The role of an operating system is essential in the context of heterogeneous and homogeneous MPSoC mainly to benefit from a large variety of services to ease the exploitation of the available resources (cooperative and pre-emptive multi-tasking, process management, multi-threading, etc.). However, its impact on the energy consumption is however non-negligible. Several researches have studied the impact of OS without actually proposing consumption models. [48] and [56] shown that the energy consumption can rise from 6% to 50% with an OS, depending on the application as well as the operating frequency and supply voltage.

Acquaviva et al. proposed in [48] a new methodology to characterise the OS energy overhead. They measured the energy consumption of the *eCos* Real Time Operating System (RTOS) running on a prototype wearable computer, HP's *SmartBadgeIII*. Then, they studied the energy impact of the RTOS both at the kernel and at the I/O driver level and they determined the key parameters affecting the energy consumption. This work studied the relation between the power and performance of the OS services and the CPU clock frequency. Acquaviva et al. performed this analysis but they did not model the energy consumption of the OS services and drivers. Development tools were proposed to analyze the OS energy overhead from simulations at the micro-architectural level like with *Simbed* in [57, 146], or at the instruction level like with *Skyeye* [74, 110]. Such approaches inherit the drawbacks of the simulation level involved (time consuming cycle level simulations, simple processor models, larger errors, etc.).

In the frame of the OPEN-PEOPLE project, the particularity of our approach is that it is based on actual measurements on the electronic boards, and that it aims at proposing consumption models for every component in the embedded systems considered. Following this direction, we propose models to take into account complete real-time embedded systems, including complex processors, reconfigurable components (FPGA), and dedicated OS services such as scheduling, context switching, or inter-process communications [81, 124].

To ensure a high level of energy and power optimization, several studies have been proposed for the exploration of scheduling policy and dynamic Voltage/Frequency management. As a first approach, authors present in [68][130] the Dynamic Power Management (DPM) techniques for multiprocessor real-time systems. These techniques improve power conservation capabilities by changing selectively the multiple *idle* states taking into account the cost of power transitions [125] [63]. They are divided into predictive schemes trying to predict future scheduling input to the system and stochastic schemes designing power management through the controlled Marcov process [62]. Bhatti [68] proposes a DPM technique called AsDPM (Assertive DPM) that allows the extraction of all idle time from some processors and clusters them on some others to elongate the duration of idle time on the target processors. This allows to reduce the transition power cost between the different states. As a second approach, the Dynamic Voltage and Frequency Scaling (DVFS) is another widely used energy reduction strategy. It allows dynamic control of voltage and frequency to reduce both dynamic and static power consumption [129]. Real-time DVFS techniques are classified into inter-task by redistributing the slack time between tasks [98] [128] or intra-task by re-

distributing the slack time inside the same task [67].

As a promising software engineering approach, MDE was used for power estimation in several design environments. In [79] and [78], an analytical method for power estimation is proposed. The power estimates are obtained by an estimation tool called SPEU (System Properties Estimation with UML). This method is based on describing an application using UML diagrams and the UML-SPT profile [111]. Using model transformations and the SPEU tool, analytical estimates can be obtained using the cost specified in the models such as the costs associated to the services of a processor. A similar approach is adopted by the CAT tool that uses the Architecture Analysis and Design Language (AADL) [43] to describe embedded applications and operating systems. In [53], an extension to the MARTE profile, with a Dynamic Power Management (DPM) profile, is proposed. This approach considers an embedded application as a set of use cases and links a power mode to each use case. In a power mode, the components of a systems may have different power states. The above presented environments, allow fast system energy/power estimation at the function level which alters the accuracy criterion. In our work, we go further in the usage of MDE by leveraging the model transformations to target multiple platforms (e.g. simulation or RTL imlementation) needed during the embedded system design. The hardware/software interaction has been kept by the means of a high level co-simulation or RTL implementation in order to extract accurately the power data. Finally, we have defined a transformation chain that turns automatically the MPSoC high level specification into an executable implementation.

3.3 The OPEN-PEOPLE platform

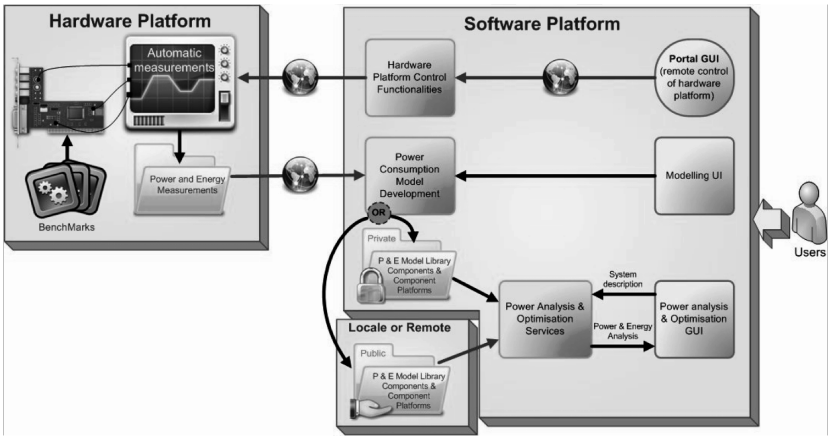

FIGURE 3.2 – Global view of the OPEN-PEOPLEplatform.

OPEN-PEOPLE stands for Open Power and Energy Optimisation PLatform and Estimator. The platform is defined for estimation and optimisation of the power and energy consumption of complex electronic systems. Among the targeted systems, we mention homogeneous and heterogeneous MPSoC based on ASIC or FPGA technology. Our platform allows power estimation using :

– direct access to the hardware execution boards and the measurement equipments. This first alternative enables designer to measure the real power dissipation of the target system. To do so, the low level description of the system (C, VHDL, etc.) is carried out

natively on the target board. Furthermore, this alternative is used to build new power models for hardware or software component as it will be described in Section 3.6. Several boards have been integrated in our automated bench and equipped with special gear to allow for power consumption measurement. Among those boards, one may find some processor based boards (OMAP 3530, OMAP L138) or some FPGA based boards (Spartan 6, Cyclone 3, Virtex 5, Virtex 2, etc.).
– a set of ESL tools coupled with accurate power models elaborated within the first alternative. Mainly, we offer tools at the functional and transactional levels in the context of multilevel exploration of new complex architectures.

The figure 3.2 presents a global view of the platform which is based on two main parts ; the software part and the hardware part. The software user interface ensures the access to the power measurements and helps the designer to define energy models for the hardware and software system components. From the measurements, the designer can build models and compute an estimation of the energy and/or power consumption of his system. In addition, from this software user interface, the hardware platform can be controlled. The hardware part consists of the embedded system boards, the measurement equipments, and the computer that controls these different elements and schedules the list of measurements required by different users.

In the frame of the OPEN-PEOPLE project [34] [33], new methods and tools to model different components of a SoC architecture were proposed including processors, hardware accelerators, memories, reconfigurable circuits, operating system services, etc. Furthermore, this project studies how the complete estimation and validation can be performed for complex systems within an acceptable simulation time. The OPEN-PEOPLE platform proposes also a set of optimisation tools at different levels of description and/or for the different target boards (architectural optimisations, operating system optimisations, etc.).

3.4 Background notions

In the following, we may refer indifferently to energy or power models, knowing that passing from one to the other only involves the actual execution time of the object considered. Power and energy consumption are equally important concerns to us : the first is directly linked to the power dissipation and operating temperature of the hardware and the second impacts on the size and lifetime of batteries.

The aim of this section is to identify the sources of power consumption in an embedded system. As shown in figure 3.3, we consider an embedded system in its entirety : the software (i.e. the application code) at the top level, the hardware (the electronic board onto which the code is running) at the lowest level, and between them the operating system (OS) and the associated services. For instance, one task obviously requests the processor, the cache, and the main memory, but also involves the process manager and the scheduler, which begets context switches and eventually more processing and memory accesses. The same task may also explicitly use Inter Process Communication (IPC) services, or need access to external peripherals. As we have seen in Section 3.2, the OS energy overhead may take a considerable part of the overall system consumption. It actually depends on the application complexity and the number of services called.

For the hardware tasks, the sources of consumption are generally different. Indeed, hardware tasks are generally data intensive and the designer normally does not use the operating

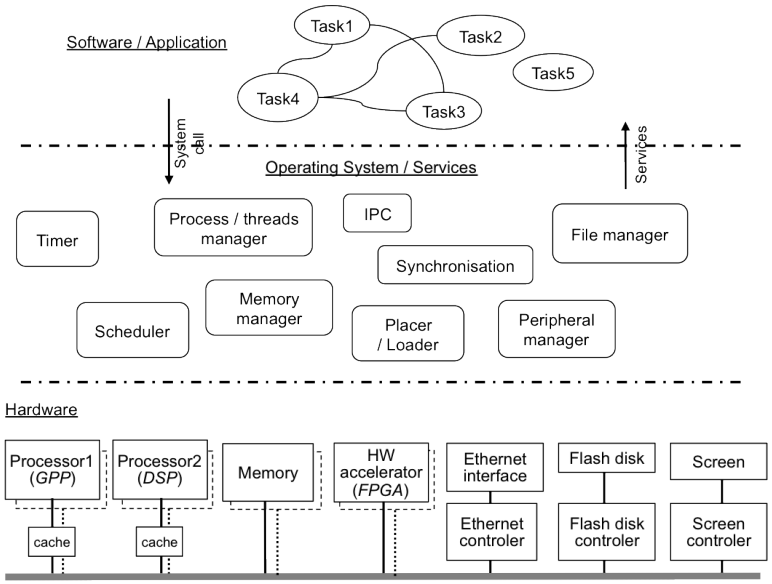

FIGURE 3.3 – View of the embedded system layers

system service calls for this type of computation. The source of consumption for a hardware task is then not linked to the operating system execution but it is related mainly to the reconfigurable logic on which it is executed. Nevertheless, each hardware task consumes and/or produces data from/to others blocks (processors, memories, I/O, etc.), so an important overhead due to data transfers can appear for these tasks. For example, when a software task running on a processor sends data to a hardware task, it can be considered as driver function call and it produces an energy overhead due to the IPC OS service call of the software task. The different contributions of the system parts in the global energy consumption are estimated with the help of the power models for different hardware and software components. These models are deduced from experimental measurements on real boards which increases the accuracy as will be detailed in Section 3.6. The class of systems considered, as shown in figure 3.3, are processor or multiprocessor based, with or without operating systems. These systems can be homogeneous or heterogeneous.

The variations of the power consumption during the execution time can be modeled as presented in figure 3.4 for software tasks running on processors, and in figure 3.5 for hardware tasks running on FPGA. Note that the energy is simply the area of every boxes on these figures. In figure 3.4, the bigger contribution is P_{ground}, which represents the power consumption of all the components when the system, without OS, is not running any application. This power consumption can be quite important especially for embedded systems on FPGA. Energy overhead of the different tasks and OS services comes in addition to this first one. More or less additional boxes may be considered depending on the actual system and the application. For the software part, we focused on the correlation between the energy consumed by the application and the services of the OS. The relationship of energy is given by equation 3.1 :

$$\forall T_i, E_{T_i} = E_{intra-task} + \sum_{1 \leq j \leq p} \alpha_{i,j} \times E_{S_j} \tag{3.1}$$

Where E_{T_i} represents the energy consumed by the task T_i, $E_{intra-task}$ is the energy consumed by the task routines and operations, p is the number of services used by the task T_i, $\alpha_{i,j}$ is the occurrence of using the service S_j on the task T_i, and E_{S_j} represents the elementary energy cost of the service S_j.

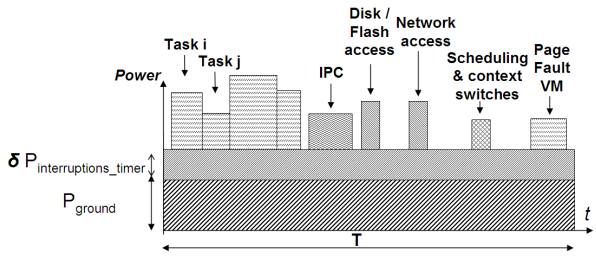

FIGURE 3.4 – Energy contribution of the software part running on GPP or DSP

For the hardware part, we refer to figure 3.5 to represent the energy contribution of the tasks placed on FPGA. Here, two P_{ground} powers are represented. The first corresponds to the power consumed by the configurable memory plan which maintains the task configurations in place during the execution, while the second represents the static power consumed by the active elements of the circuit (i.e. the static power for the configurable logic elements, the digital signal processing blocks, BRAM memories, interconnects, etc.). As shown in the figure, for each task configured (or reconfigured) in the configurable space, an additional energy is necessary to load the bitstream within the configurable memory plan. Note that figure 3.5 illustrates the partial reconfiguration paradigm with the possibility to configure a specific part of the circuit while the static part continues to operate without interruption. In this figure, we illustrated the initial configurations for tasks $HTask_i$, $HTask_j$ and $HTask_k$ which induce additional energies that depend mainly on the related logic configurations. We also illustrated a specific scenario where the task $HTask_i$ is reconfigured dynamically. Figure 3.5 also shows the placer/loader activities to manage the reconfiguration process. For each reconfiguration, the placer/loader service is called, and the first step consists in finding a sufficient area on the reconfigurable area, the placer supports this job. The second step consists in loading the bitstream within the reconfigurable memory, this step is supported by the loader. As illustrated in figure 3.5, for each reconfiguration, the placer is always executed, but the loader is optional when the task is already configured in the reconfigurable area.

For the reconfigurable space, if we consider data intensive computation tasks which are not preemptable (to ensure a high performance execution) and without operating system service calls, the model of energy consumption can be defined as follows :

- Es_{ground} is the static energy consumed by the configurable memory plan ;
- Ea_{ground} is the static energy consumed by the active elements of the global FPGA circuit. Even if these elements are not configured, a static energy is consumed by this part. We can note that some FPGA circuits provide some mechanisms to reduce this static power significantly ;
- Et_i is the energy consumed by the hardware task $HTask_i$ during its execution. This energy must represent the consumption of the task with the corresponding data transfers. For more flexibility, it is possible to provide several implementations for each task with different performance/power trade-offs ;
- $Econf_i$ is the energy necessary to configure the task $HTask_i$ for its first execution. For

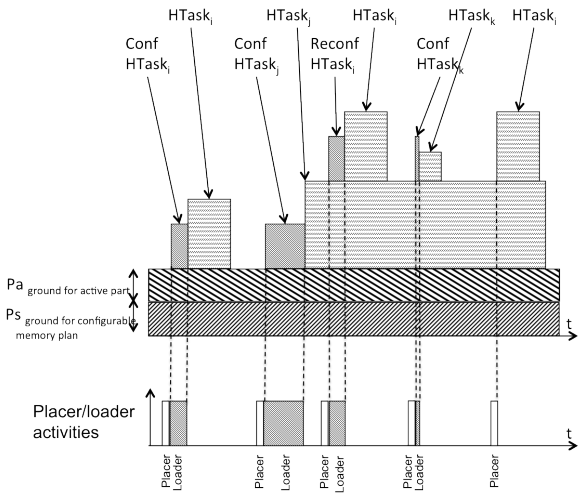

FIGURE 3.5 – Energy contribution of the hardware part running on a FPGA circuit

this step, the operating system must manage the configuration, and this is ensured by the operating system service `Placer/Loader` in figure 3.3. The execution of this service leads to active the `file manager` service of the OS and thus leads to consume an important energy to access to the bitstream file ; figure 3.5 does not show this energy contribution generated by the OS calls, but this energy is included in the OS contribution.

– $Erecon f_i$ is the energy necessary to reconfigure the task $HTask_i$ for the other executions. If we consider that the tasks are not preemptable, the context of the tasks does not need to be stored and in this case $Erecon f_i = Econ f_i$.

Finally, the global energy is defined as :

$$\begin{aligned} E_{fpga} &= Es_{ground} + Ea_{ground} \\ &\quad + \sum_{i=1}^{Nht} (Econ f_i + Et_i) \\ &\quad + \sum_{i=1}^{Nht} \sum_{j=2}^{Ne_i - 1} \left(Erecon f_i * \beta_{i,j} + Et_i \right) \end{aligned} \tag{3.2}$$

With Nht is the number of tasks to be executed in the reconfigurable space, Ne_i is the number of executions for the task $HTask_i$ and $\beta_{i,j}$ is a binary variable equal to 1 if the task $HTask_i$ must be reconfigured for a new execution or equal to 0 if the task $HTask_i$ is already configured and just need to be launched. The value $\beta_{i,j}$ depends on the execution order of the tasks, which is dynamically decided (on-line) by the operating system.

3.5 Power-aware design methodology :

This section details our power-aware design methodology for MPSoC to cover several design layers. The objective is to offer for each step a power estimation tool in order to have a gradual refinement of the design space solution based on the power or energy criteria.

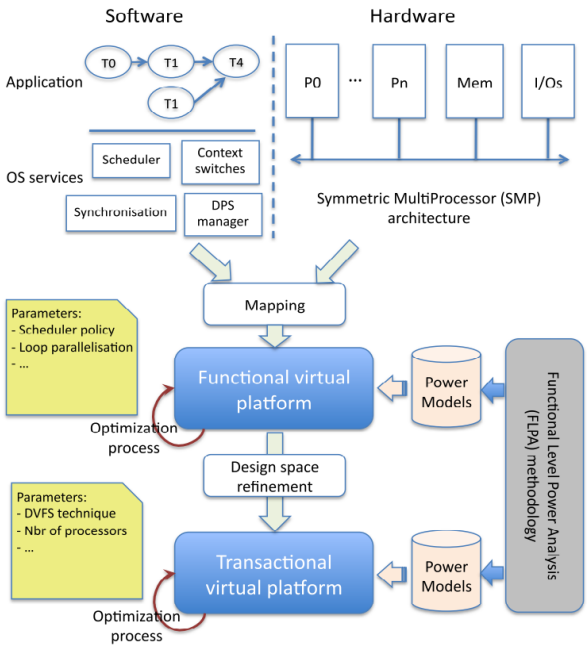

FIGURE 3.6 – A seamless multi-level power-aware design methodology

In order to cope with the design complexity, we focus specially on the functional and the transactional levels that offer different trade-offs between accuracy and estimation time as depicted in figure 3.6. For each level, several power models are developed for consumption estimation and optimisation taking into account all the embedded system relevant aspects ; the software, the hardware, and the operating system. The integration of runtime power management techniques is another benefit of our methodology.

3.5.1 Functional-level power estimation

To estimate the energy consumption of an application running on a hardware platform, it is necessary to characterise the power/energy consumption of the target by identifying the typical application tasks that will be executed on the system. At the functional level, to achieve simulations of the application, we use a multiprocessor simulation tool named STORM [45] (Simulation TOol for Real-time Multiprocessor scheduling). The input of this tool is the specifications of the hardware and software architectures together with the scheduling policy ; it simulates the system behavior using various task's characteristics (task execution time, processor functioning conditions, etc.) in order to obtain the chronological track of all the scheduling events that occurred at runtime, and computes various real-time metrics in order to analyse the system behavior and performances from various point of views. However, STORM executes the application tasks on the host machine without referring to the micro-architecture of the target embedded processor. For this reason, the power/energy characterisation is performed at the task level where the power consumption is coarsely considered. Such approach has the advantage of high speed simulation. We highlight that STORM is an example of simulation tool that can be used at the functional level. In our previous work [16], we succeeded to plug the SoftExplorer tool [81] as a part of our multilevel power aware design methodology as it will be illustrated in Section 3.7.

At the functional level, we focus on the correlation between the energy consumed by the application and the services of the OS. The relationship of energy is given by equation 3.1. At the functional level, the $E_{intra-task}$ is characterized approximately according to the selected processor and the operating frequency which can decrease the estimation accuracy. This entity will be detailed with fine-grain parameters at the transactional level in order to increase accuracy as it will be presented in the next subsection.

For the second part of the equation, we perform bench tests that stimulate each OS service in order to characterise the related energy cost. Then, real-board experiments are conducted to measure the variation of the energy consumption depending on the parameters that we change. The obtained power model will be integrated in the STORM tool in order to estimate the energy consumption of the application.

3.5.2 Transactional-level power estimation

As stated before, the power/energy estimation at the functional level is performed without referring to a real micro-architecture description. In the second step of the power-aware design methodology, we will rely on a transactional simulation in order to carry out the application tasks using an Instruction Set Simulator (ISS) considering the OS services. Our objective is to refer to the occurrences of the micro-architectural activities in order to estimate the intra-task power consumption accurately. In addition, the power consumption of the OS services (E_{S_j}) will be also considered which leads to better precision.

To do so, we define the architecture of a hybrid power estimator [28] [29] that includes a fast transactional SystemC simulator plugged with a *transactional power estimator* as shown in figure 3.7. The transactional power estimator evaluates the consumption of the target system with the help of the elaborated power models. It takes into account the architectural parameters (e.g. the frequency, the number of processors, the processor cache configuration, etc.) and the application mapping. It also requires the different activity values on which the power models rely. In order to collect accurately the needed occurrences, the power estimator communicates with a fast SystemC simulator at a TLM level. The combination of the transactional power estimator with the FLPA power modeling methodology leads to a hybrid approach that gives a better trade-off between accuracy and speed in comparison with the functional level.

The vital function of system-level power estimation is to offer a detailed power analysis by the means of a complete simulation of the application. This process is initiated by the functional power estimator through *the data and task interface* (figure 3.7). In this way, the mapping information is transmitted to the fast TLM SystemC simulator. Our simulator consists of several hardware components which are instantiated from the SoCLib [121] library in order to build a virtual prototype of the target system. We highlight that processors are described using ISS that sequentially executes the instructions. In our previous framework [55], we presented an accurate TLM simulation technique that allows to evaluate the MPSoC performances. In the power estimation step, the simulator collects the activities that are influenced by the application and the input data. At the end of the simulation, the values of the activities are transmitted to the power consumption models or power estimator kernel using *the activity counter interface* in order to calculate the global power consumption as illustrated in figure 3.7.

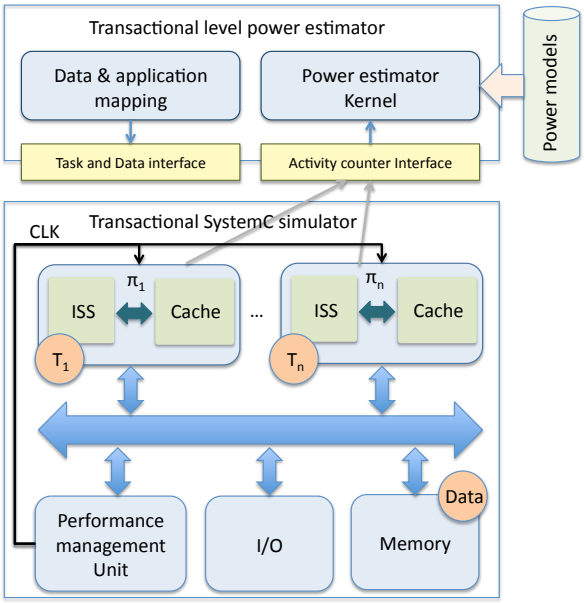

FIGURE 3.7 – Transactional power estimator tool functioning

3.5.3 Optimisation process

The optimisation strategy in our power-aware design methodology is introduced at different levels in order to have a gradual refinement of the design space. It is performed first off-line by the mean of design space exploration and at runtime using dynamic power management techniques.

For the functional level, the off-line optimization process starts by dividing the architecture into different functional blocks. Each block represents a unit ensuring a specific functionality such as the memory unit, the processing unit, etc. Then, power consumption of each block is modelled using simulations or measurements. Afterwards, functional blocks influencing the processor power consumption are identified. Finally, several algorithmic and architectural parameters are varied to reduce the power consumption.

At the transactional level, the off-line optimization process consists in exploring several architectural solutions (type of processor, number of processors, ...) in order to tune the hardware according to the application needs [30] [27]. Using different available Intellectual Properties (IPs), several simulations are carried out yielding execution time, power and energy estimations. Energy optimization in embedded systems can be performed at runtime. It could be solved at the system level while scheduling the application tasks. In this context, different power and energy management techniques are proposed to optimise and reduce the energy consumption while taking into account the features offered by the target platform [24].

3.6 Consumption modeling / power models

This section concerns the power model elaboration in order to consider the contribution of each part of the system on the total power consumption. Our objective is first to develop a set of power models for each part and second to plug them into our design tools at the

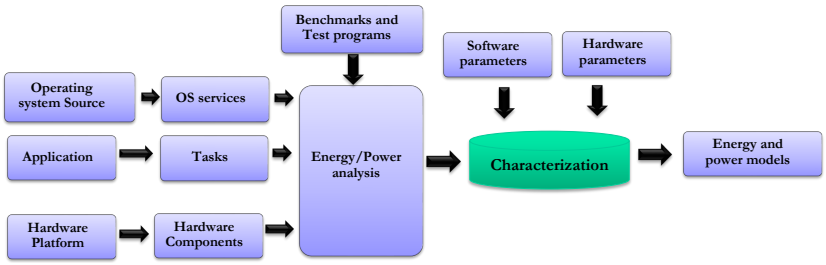

FIGURE 3.8 – System power modeling process

functional and transactional levels. In our framework, the FLPA methodology is used to develop generic power models for different target platforms. As depicted in figure 3.8, the proposed method aims to extract power/energy models of embedded OS services, software application and hardware components.

In the energy analysis step, various hardware and software parameters which influence the energy consumption are identified and then energy profiles are traced according to the variation of these parameters. From the energy traces, a curve fitting of the graphical representation will allow us to determine the power consumption models by regression. The obtained power models are expressed in the form of analytical equations or table of values. This approach was proven to be fast and precise [123]. The main advantage of this methodology is to obtain models which rely on the functional parameters of the system with a reduced number of experiments. As explained in the previous section, FLPA comes with few consumption laws, which are associated to the consumption activity values of the main functional blocks of the system. The generated power models have been adapted to system-level design, as the required activities can be obtained from a system level (functional or transactional) environment.

3.6.1 Power modeling of OS services

In this section, we will illustrate two examples for OS service power modeling : the context switch and the scheduling. Additional power models for other OS services can be found in [16].

3.6.1.1 The context switch

Context switch is a mechanism which occurs when the kernel changes the control of the processor from an executing process to another that is ready to run. The kernel saves the state of current process including the processor register values and other data that describes this state. Then, it loads the saved state of the new process for execution.

Context switching introduces direct and indirect power overheads [107]. Direct context switch overheads are related to saving and restoring processor registers and flushing the processor pipeline. Indirect overheads involve the switch of the address translation maps used by the processor when the threads have different virtual address spaces. In addition, when a thread T_1 is switched out and a new thread T_2 starts the execution, the cache state of T_1 is perturbed and some cache blocks are replaced. So, when T_1 resumes the execution and restores the cache state, it gets cache misses. The OS memory paging represents also a

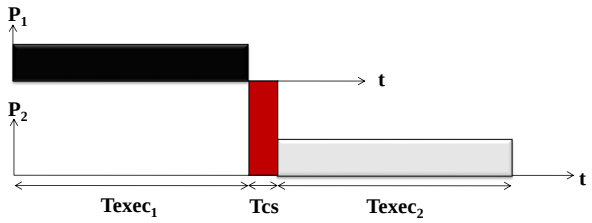

FIGURE 3.9 – Extraction of the context-switch time : Scenario1

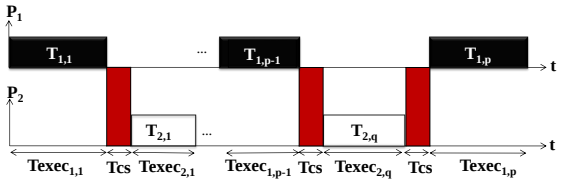

FIGURE 3.10 – Extraction of the context-switch time : Scenario2

source of the indirect overhead since the context switch can occur in a memory page moved to the disk when there is no free memory. Prior research has shown that indirect context switch overheads [137], mainly the cache perturbation effect, are significantly larger than direct overheads. Therefore, this work focuses only on indirect overhead of context switch.

To characterise the energy consumption of the indirect context switch, we create a set of threads in a multitasking environment using the POSIX standard. The used platform is the Cortex A8 processor-based OMAP3530 running a 2.6.3 Linux OS version. The benchmark application is composed of two threads P1 and P2. In order to extract the consumption of context switch, the application was executed using two scenarios. In the first scenario, only one context-switch is generated between P1 and P2 as depicted in figure 3.9. In the second scenario, which is depicted in figure 3.10, n context-switches are generated. Tcs represents the time of the context switch, and $T_{i,j}$ the execution time of the j-th job of the process P_i.

The total execution times of scenario1 and scenario2 are respectively $T_{scenario1}$ and $T_{scenario2}$. They are presented by equations 3.3 and 3.4 :

$$T_{scenario1} = Texec_1 + Tcs + Texec_2 \tag{3.3}$$

$$T_{scenario2} = \sum_{1 \leq i \leq p} T_{1,i} + \sum_{1 \leq j \leq q} T_{2,j} + (n \times Tcs) \tag{3.4}$$

Where p and q represent respectively the number of jobs of P_1 and P_2.
The context switch time Tcs and the context switch power overhead Pcs are calculated following the equations 3.5 and 3.6 :

$$Tcs = (T_{scenario2} - T_{scenario1})/(n-1) \tag{3.5}$$

$$Pcs = (P_{scenario2} - P_{scenario1})/(n-1) \tag{3.6}$$

The context switch energy overhead is computed as :

$$Ecs = (P_{scenario2} * T_{scenario2} - P_{scenario1} * T_{scenario1})/(n-1) \tag{3.7}$$

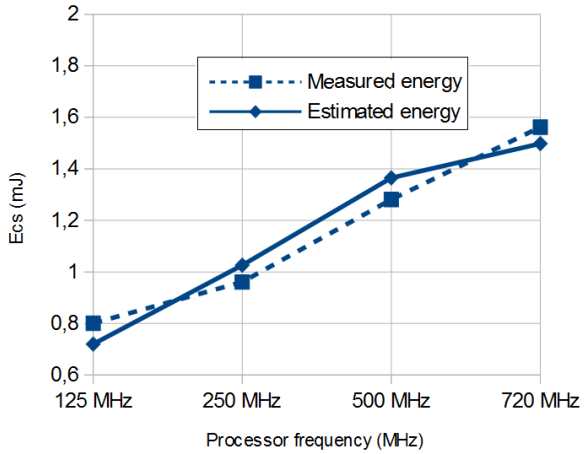

FIGURE 3.11 – Context switch energy variation as a function of CPU frequency

Where $P_{scenario1}$ and $P_{scenario2}$ are respectively the average power consumption of the benchmarks in scenario1 and scenario2.
In order to study the impact of the CPU frequency on the energy overhead of the context-switch, the two scenarios presented above are re-executed using different frequencies. Two cases are considered : static and dynamic CPU frequencies.

► **Static frequency case** The two scenarios of the benchmark are executed for different CPU frequencies. The frequency remains static during a scenario execution. As a result, the context switch energy follows the law presented in the equation 3.8 :

$$Ecs(f) = 1.28 \times 10^{-3} \times f + 0.641 \tag{3.8}$$

where f is the CPU frequency. The units of Ecs and f are respectively mJ and MHz. Figure 3.11 plots the context switch energy overhead as a function of the frequency. The average error of the proposed methodology results against the physical measurements is about 3.4%.

► **Dynamic frequency case** The core frequency is dynamically changed during the two scenarios of the benchmarks. Processes P1 and P2 are executed at a frequency $F1$ and $F2$ respectively. When the processor preempts the process P1 and executes the process P2, the core frequency changes from $F1$ to $F2$ and vice-versa. Figure 3.12 illustrates the context switch energy variation according to changing the CPU frequency. Actually, for the processor core, a set of voltage and frequency couples is specified, named operating points. Running on high frequency requires also high voltage and vice versa. For raising the frequency and supply voltage, the microprocessor sets a new VID (voltage identifier) code to have a higher output voltage than the current one, and conversely. This operation leads to time and energy overhead [116]. Also, the more the difference between $F1$ and $F2$ is, the higher context switch energy is. This is due to the perturbation of the processor's cache memory resulting from the variation of processor bus frequency which varies with the processor frequency.

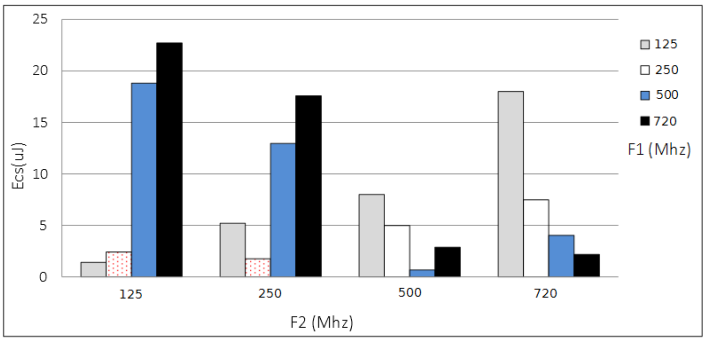

FIGURE 3.12 – Context switch energy variation as a function of dynamic CPU frequency scaling

3.6.1.2 The scheduling routines

Scheduling routines and operations could generate power overhead on the processor and/or memory components. They are considered as system calls and only consist in switching the processor from unprivileged user mode to a privileged kernel mode. To quantify power and energy overhead of embedded OS scheduler routines and operations, we have to build test programs containing threads with different priorities, we measure in the first step the average energy consumed by the standalone tasks without scheduling routines, and then with scheduling routines.
$E_{Scheduling}$ represents the energy consumed by the scheduling operations. It is calculated as showed in equation 3.9 :

$$E_{Scheduling} = E_{withsch} - E_{withoutsch} \tag{3.9}$$

Where $E_{withsch}$ and $E_{withoutsch}$ represent respectively the energy consumed by the benchmarks with and without scheduling routines. In order to study the impact of the CPU frequency on the scheduling overhead, we varied the frequency considering the operating points of the processor (125 MHz, 250 MHz, 500 MHz and 720 MHz). The used application scenario is composed of 10 processes scheduled using the SCHED_OTHER policy. This policy is based on dynamic priorities of processes in order to ensure a fair progress of the scheduling. Figure 3.13 depicts the variation of the energy consumption of scheduling routines with the processor frequency. The energy consumption law for the scheduling routines is formulated in equation 3.10. The average estimation error is around 0.35%. The obtained decreasing curve can be explained as follows. When running this experiments, the steady state current (and hence the power) profile obtained is almost flat since the processor does not access the external bus. Consequently, the energy cost of the scheduler is proportional to the execution time of the scheduling routines. Therefore, it decreases with the frequency increase.

$$E_{scheduling}(f) = -59.649 \times 10^{-3} \times f + 3.106 \times 10^{2} \tag{3.10}$$

3.6.2 Transactional power modeling

As explained previously, the transactional level power estimation refers to the occurrences of the micro-architectural activities. Therefore, it gives a more accurate estimation of

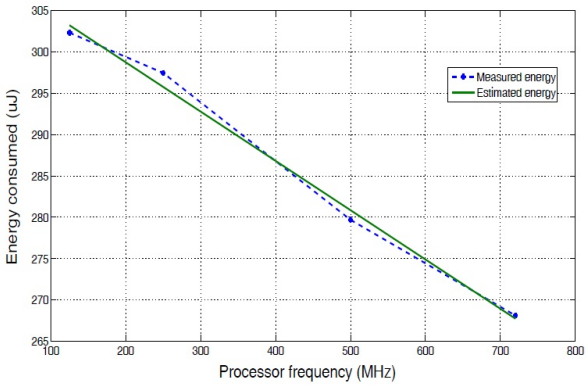

FIGURE 3.13 – Scheduling routines energy variation as a function of CPU frequency

OS service and intra-task consumptions. From a power consumption point of view, OS service at the transactional level can be divided into services which involve micro-architectural activities (task switching, scheduling, etc.) and the other services (frequency switching). At transactional level, we use the same power model as the functional level for frequency switching overhead. As for the other consumptions (IPC, scheduling and intra-task), we do not consider them separately. We rather consider the micro-architectural activities that they all involve such as instruction execution and memory access. In order to characterise these activities, the first step is to divide the architecture into different functional blocks and then to cluster the components that are concurrently activated when the code is running. There are two types of parameters : 1) *algorithmic parameters*, which depend on the executed algorithm typically the cache miss or instruction per cycle rates, and 2) *architectural parameters*, which depend on the component configuration set by the designer, typically the clock frequency. For instance, Table 3.1 presents the common set of parameters of our generic power model. These sets of parameters are defined for a general class of RISC processors. Additional parameters can be identified for specific processors based-architecture such as Superscaler. The second step is the characterisation of the embedded system power consumption when the parameters vary. These variations are obtained by using some elementary assembly programs (called scenario) or built in test vectors elaborated to stimulate each block separately.

In order to prove the usefulness and the effectiveness of the proposed power estimation methodology, we used an ARM Cortex A8-based architecture implemented into the OMAP3530 platform. The OMAP3530 contains an ARM Cortex A8 processor (16KB, 2-way set associative instruction and data caches and 256KB L2 cache). The processor has access to the off-chip memory (SDRAM) via the processor bus interconnect. As explained above, we used the FLPA methodology to generate a power model for the target system. As a first step, we divided the architecture into different functional blocks such as the core clock system, the memory system, and the functional unit for ARM Cortex A8 processor as shown in the figure 3.14. A parameter is denoted for each functional block such as $\gamma 1$ and $\gamma 2$ respectively for L1 cache miss rate and L2 cache miss rate. The second step is the characterisation of the power model by varying the parameters. In our work, characterisation is performed by measurements on the real board.

Equation 3.11 shows the power consumption model for the ARM Cortex A8 processor using the FLPA methodology. The input parameters are the processor frequency ($F_{processor}$(MHz)), the Instruction Per Cycle (IPC), and the cache miss rate ($0 < \gamma < 100\%$).

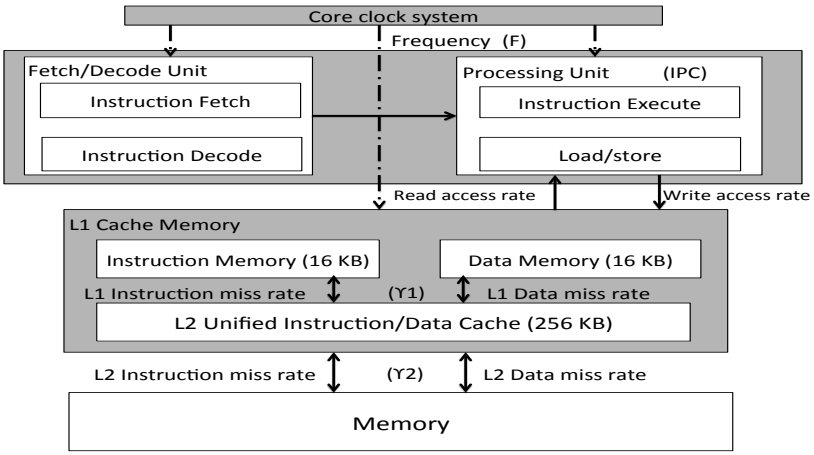

FIGURE 3.14 – Main functional blocks of ARM Cortex A8 processor

TABLE 3.1 – Generic power model parameters

Algorithmic	Name	Description
	τ	External memory access rate
	γ	Cache miss rate
	IPC	Instruction per cycle rate
Architectural	$F_{processor}$	Frequency of the processor
	F_{bus}	Frequency of the bus

The system designer chooses the frequency of the processor while the cache miss rate and the IPC are considered as an activity of the processor, which could be extracted from the simulation environment. Additional power models for other processors such as the dual core ARM Cortex A9 are given in [30].

$$P(mW) = 0.79F_{Processor} + 18.65IPC + 0.26(\gamma 1 + \gamma 2) + 10.13 \tag{3.11}$$

3.6.3 FPGA power model

A power model has been built for the reconfigurable part of the Virtex II FPGA. This model has been built with coarser granularity to be adequate at the system level as stated before. This model does not come as a multi-linear equation of the frequency F, switching activity β and area utilization α. For this reason, a 3 entries table of consumption values is used. The power is estimated by interpolation of these 3 input parameters. Figure 3.15 illustrates the variation of the FPGA power consumption according to area utilization and the switching activity with an operating frequency set to 100 MHz. The same power modeling methodology can be applied for different FPGA circuits.

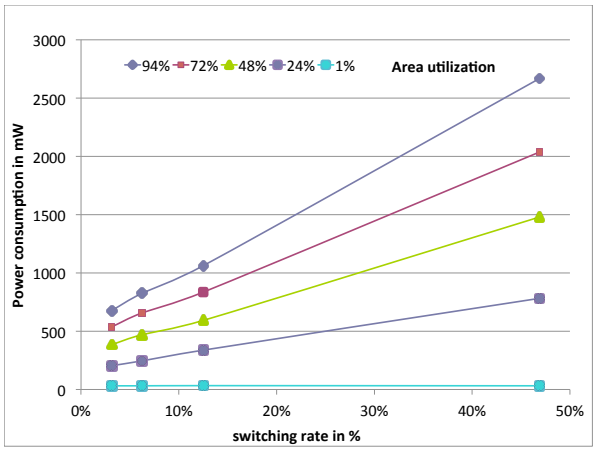

FIGURE 3.15 – FPGA power consumption with 100 MHz frequency

3.7 The multi-level design space exploration :

3.7.1 The JPEG case-study

This section describes the usefulness and the effectiveness of our power estimation methodology for a PowerPC 405-based SoC implemented on the Xilinx Virtex II Pro FPGA (XupV2Pro) platform. The Virtex II Pro FPGA contains two hardware PowerPC 405 processors that have a 16KB, 2-way set associative instruction and data caches. In addition, a large number of configurable logic blocks (CLB) are available for implementing hardware accelerators. Each processor has the access to the on-chip memory (BRAM) and the off-chip memory (SDRAM) via Processor Local Bus (PLB). Different power models are developed and integrated into system level design tools at the functional and the transactional level as explained in Section 3.5 [29]. At the functional level, we used the SoftExplorer tool for power estimation. Certainly, XupV2Pro platform is considered as an old technology. However, the same methodology can be applied for recent technology such as the Zynq platform that embeds a dual core ARM Cortex A9 processor and tens of thousands of programmable gate arrays on the same chip. We used the JPEG (Joint Photographic Experts Group) application as a benchmark. The JPEG application consists of 6 main tasks : acquisition of the input image, conversion RGB (Red, Green and Blue) to YUV (luminance, blue chrominance, and red chrominance components), Discrete Cosine Transform (DCT), Quantization, Huffman coding, and rebuild of the output image.

3.7.1.1 Monoprocessor architecture

As a first scenario, we used the JPEG application with a PowerPC monoprocessor based architecture. To do so, we developed a system level prototype of the PowerPC based SoC, with the help of SystemC models including ISS for the target processor, with the cache parameters and bus latencies set to emulate the real platform behaviour. A set of counters are injected into the simulator to determine the values of different miss rates : read data miss, write data miss and read instruction miss of the corresponding caches. The fast SystemC simulator takes the full JPEG application with the real standard frame size of 256×256 pixels and simulates it entirely in order to collect the required activities.

TABLE 3.2 – Application miss rates

Program	Instruction miss rate (%)	Read Miss rate (%)	Write Miss Rate (%)	Total Miss Rate (%)
acquisition	0.003386	3.56	31.73	0.02
rgb2yuv	0.001128	3.03	99.91	5.64
dct y	0.002283	4.49	40.72	3.88
dct u	0.000315	4.49	40.72	3.88
dct v	0.000314	4.49	40.72	3.88
qt y	0.000812	2.06	99.88	5.58
qt u	0.000406	2.06	99.93	5.58
qt v	0.000406	2.06	99.94	5.58
huff y	0.004375	4.58	20.11	0.85
huff u	0.000515	4.57	19.8	0.84
huff v	0.000643	4.56	19.61	0.84
rebuild image	0.298380	3.05	25.19	2.87
complete application	0.000012	0.029	0.09	0.012

Table 3.2 shows the detailed activities of each task in the application, as a result of the SystemC simulation. From these results several remarks can be drawn. First, we can notice that instruction cache miss rates and read data miss rates are very low when compared with write data miss rates. This is due to the fact that the task kernel is small (a small number of instructions) and that the volume of data accessed is also small compared to the cache size (16 KB). With the new sub-micron technologies however, static power consumption cannot be neglected. For this reason, some software processors, such as the Microblaze, come with reconfigurable cache sizes to fit the application requirements. Secondly, we observe that the data write miss rates have a high impact on the total power consumption. This is due to the algorithm structure which does not favour the reuse of data output arrays, and to the usage of write-through cache policy. As we can see, the statistics collected in Table 3.2 can help in tuning the application structure for a better optimization of the system power consumption.

In the next step, using the obtained results and the developed power models for the platform [29], we estimated the total power consumption of each task. Figure 3.16 illustrates the results and shows the comparison between the proposed hybrid estimator, the SoftExplorer tool introduced in Section 3.2, and the real board measurements. Negative and positive errors correspond respectively to under and over estimation of the consumption. The average error is the mean value of the absolute values of those errors. First, the hybrid power estimator has a negligible average error equal to 0.02% which offers better accuracy than SoftExplorer with its average error of (3.32%). Indeed, the activities captured in the SystemC simulator are more accurate than the static analysis or rapid profiling of the code performed by SoftExplorer. The most of error is reported from the acquisition and rebuild tasks, respectively 22.5% and 6.36%. In fact, these two tasks use operating system calls to read and write from/to files. Those system calls are however only executed by the system level simulator by means of a virtual file system, which does not reflect precisely the real operating system behaviour. Finally, without considering the acquisition and rebuild tasks, the hybrid estimator gives an average error of 1.32% while SoftExplorer is 3.17%.

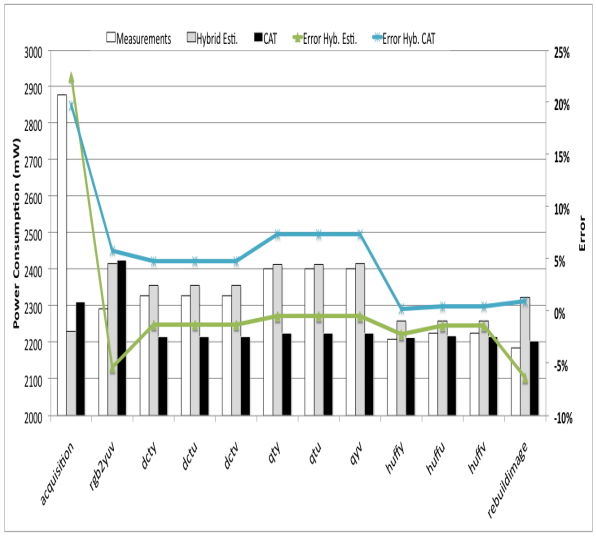

FIGURE 3.16 – Power Estimation and Comparison with SoftExplorer

3.7.1.2 Homogeneous multiprocessor architecture

The second scenario involves an homogeneous architecture with identical processors to run the JPEG application. To evaluate the impact of the number of processors on the execution time and total energy consumption, we executed the JPEG on systems with 1 to 8 processors. The PowerPC frequency was set to 300 MHz and the PLB frequency to 100 MHz. All the processors execute the same workload but on different image macroblocks. Figure 3.17 reports the execution time in ms and the total energy consumption in mJ.

Given these results, we see that adding processors to the system decreases the execution time, which improves the system performance. This variation is not linear because the processors share resources, which generates conflicts at some times, and reduces the speedup as waiting cycles are added to the processors execution. In terms of energy consumption, we observe that until a certain number of processors, the total system energy consumption decreases as the execution time is reduced. Adding more processors increases the power consumption, however with not the same slope as the time decreases. As we are using only the ASIC PowerPC processors integrated in the Xilinx Virtex II FPGA and the processors are executing the same workload in parallel, the static power is not influencing significantly the total consumption. But increasing the number of processors over a certain limit tends to be ineffective, as it just adds new conflicts at the PLB level, leading to more waiting cycles.

3.7.1.3 Hardware accelerator design

In this part, we emphasize the benefit of our estimation methodology in the context of heterogeneous architecture. In general, the choice of a hardware accelerator is driven principally by the performance requirements of the application and the processor usage of each task. For the JPEG application, the DCT task is the most time consuming task. Thus, it is selected to be implemented as a hardware accelerator. Various trade-offs can be done between the amount of consumed hardware resources (i.e. : the area utilization), the execution time, and the power consumption. The DCT task is highly regular and has large repetition spaces

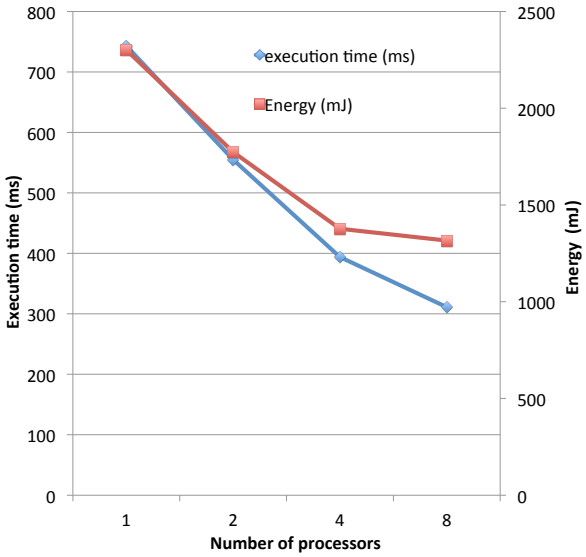

FIGURE 3.17 – Execution time and energy variation in terms of the number of processors

in its multiple hierarchical levels. Such large repetition spaces allow us to fully exploit the existing partitioning in VHDL (i.e. hardware-software and parallel-sequential hardware). System-level architecture synthesis tool such as GAUT [92] or ROCCC [141] can be used to obtain several implementations of the hardware accelerator with different trade-offs between the execution time or the number of resources [21]. Certainly, more accurate estimation of these parameters can be obtained at lower levels using the commercial RTL tools but at the price of significant evaluation time. We selected a configuration which is about 200 times faster than a software execution with a PowerPC processor running at 100 MHz. A hardware synthesis of this configuration occupies 18% of the XupV2Pro. According to the FPGA power model, the power consumption of the chosen DCT hardware accelerator is around 300mW offering 40% of power saving compared to the software execution.

3.7.1.4 Extrapolation for complete MPSoC architecture

The above developed power models will be used in the frame of system level estimation of heterogeneous MPSoC that may contain several processors and hardware accelerators. This approach is mandatory in the design flow for two reasons, even if the corresponding estimates are less accurate than those provided by real board measurements. First, system level estimation can be achieved with acceptable accuracy 10-1000x faster than the physical level taking into account the required design time. Second, it allows exploring architectures that cannot be implemented due to the hardware resource limitation or the unavailability of the target component. For instance, we cannot exceed two PowerPC based architecture using our XupV2Pro platform. Thus, it is important to have a scalable approach to address the complex system power/energy estimation issue. The equation 3.12 will be considered for the total system energy estimation. We find there the sum of the energy consumptions of every software tasks with the related operating system energy overhead (see equation 3.1) and the sum of the energy consumptions of every hardware tasks (see equation 3.2). The

consumption of the synchronization part required to access the shared resources is included in E_{OS_τ}.

$$E_{total} = \sum (E_\tau + E_{OS_\tau}) + \sum E_{fpga} \tag{3.12}$$

In our XupV2Pro platform, a software synchronization between several tasks running on different processors or hardware accelerators will call for a hardware mutex through an OS service. Several experiments have been conducted to evaluate the additional power cost of this hardware component. This study includes three parameters which are the number of masters, the processor frequency, and bus frequency as detailed in [16].

3.7.2 The H.264 decoder case-study

This section presents the experimental results that evaluate the effectiveness of our energy/power-aware design methodology and of the overall framework for power estimation and optimization. The case study application is a H.264 video decoder. The proposed multi-level power-aware design methodology was used for design exploration under a QoS constraint of 15 frames per second. In this case-study, we will use a DVFS algorithm called Dynamic Slack Reclamation (DSR) in order to optimise the power consumption at runtime. This algorithm was introduced in [24].

3.7.2.1 Application description

The case-study H.264 video decoder application is based on a high quality video compression algorithm. It relies on efficient strategies extracting spatial (within a frame) and temporal dependencies (between frames). This application is characterized by a flexible coding, high compression and high quality resolution. Moreover, it is a promising standard for embedded devices.

As shown in Figure. 3.18, the main steps of the H.264 decoding process consist in the following : A compressed bit stream coming from the Network application layer (NAL) is received at the input of the decoder. This bitstream is an encapsulation of the video frames data and parameters in form of NAL units suitable for packet transmission over the network. A video frame can be decomposed into a number of slices (2, 4, 8, etc.). The encoded data in the bitstream are obtained by encoding the slices of each frame of a video sequence. Each encoded slice is composed of one or more 16x16 macroblocks. The first step of the decoder is the entropy decoding. For each macroblock, the output of the entropy decoder is a set of quantized coefficients.

These coefficients are then inverse quantized and inverse transformed. Thereafter, the data obtained are added to the predicted data from the previous frames depending upon the header information. Finally the original macroblock is obtained after the deblocking filter, which compensates the block artifacts effect.

The H.264 video decoder application can be split into four main tasks. The *NEW_FRAME* task extracts frames from the input bitstream file. The *NAL_DISPATCH* task decomposed the frame into slices. The *SLICE_PROCESSING* task decodes the slices (entropy decoding, inverse quantization, inverse transformation, etc.). The *REBUILD_FRAME* task rebuilds the frame from the decoded slices. In this case study, we consider a video sequence where each frame is composed of 4 slices.

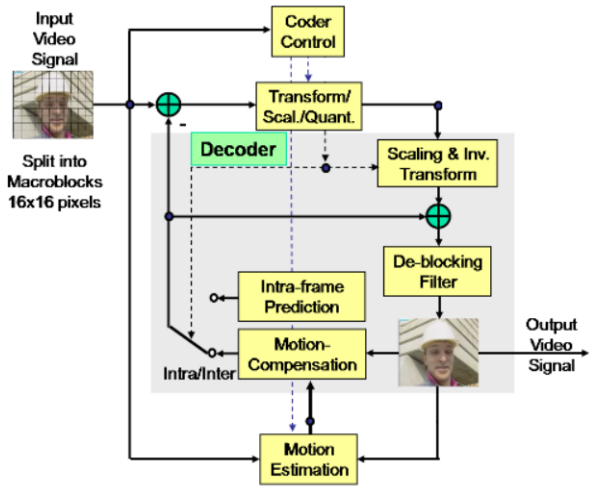

FIGURE 3.18 – Block diagram of H.264 video decoder

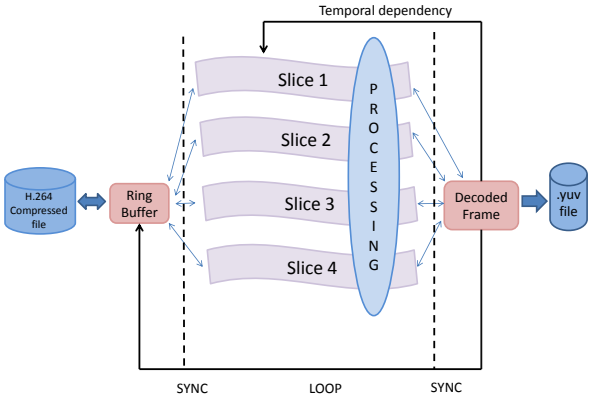

FIGURE 3.19 – Parallelization of the H.264 video decoder

Therefore, the *SLICE_PROCESSING* task can be split into 4 independent sub-tasks $SLICE\_PROCESSING_i, i \in [1..4]$, where each sub-task processes a slice of the frame. These sub-tasks can thus be executed in parallel by processors.

At the beginning of each new frame, the $SLICE\_PROCESSING_i$ sub-tasks can access the input data buffer only sequentially. Therefore, the *NAL_DISPATCH* task, which provides access to the shared resource, is protected by a semaphore. Then, $SLICE\_PROCESSING_i$ tasks are launched simultaneously. Due to temporal dependencies between frames, it is not possible to compute the next frame if the previous one has not been completely decoded. Thus, at the end of each slice computation, tasks need to be resynchronised using task named SYNC before running the *REBUILD_FRAME* task as shown in Figure 3.19.

The target architectures are Shared-Memory Multiprocessor systems. In these architectures, two or more identical processors are connected to a single shared main memory, have full access to all I/O devices, and are controlled by a single OS instance. The processors are treated equally, with none being reserved for special purposes. Each processor executes different tasks and is able to share common resources (memory, I/O device, interrupt sys-

TABLE 3.3 – Voltage-frequency levels of the ARM Cortex A8 processor

Parameter	OP 1	OP 2	OP 3	OP 5
Frequency (Mhz)	125	250	500	720
Voltage (V)	0.975	1.05	1.2	1.35
Running power (mW)	57	130	303	550
Idle power (mW)	4	7	16	28

TABLE 3.4 – H.264 video decoder application tasks characterisation

Task name	**WCET (ms)**	**BCET (ms)**	**Period (ms)**	**Deadline (ms)**	**Activation date (ms)**
New_frame	1	1	19	19	0
Nal_dispatch	2	1	5	5	0
*Slice*1*_processing*	42	21	66	66	0
*Slice*2*_processing*	42	21	66	66	1
*Slice*3*_processing*	42	21	66	66	2
*Slice*4*_processing*	42	21	66	66	3
Rebuild_frame	2	1	66	66	66

tem and so on) with other processors. These processors are connected to each other using a system bus.

3.7.2.2 Functional-level energy/power analysis

As explained previously, the energy consumption at the functional level takes into account the consumption of the OS calls as well as the intra-task consumption. For the OS services consumption, the energy models developed using the OMAP3530 platform (Section 3.6.1) were integrated in the STORM simulator together with the DSR algorithm, in order to optimise power consumption at runtime. At the functional level, intra-task consumption is estimated coarsely using only two power consumption values corresponding to the running and idle states of the processor for different frequency-voltage couples. Table 3.3 gives the power consumption of the ARM Cortex A8 processor for all possible operating points.
Intra-task consumption is determined using the following formula :

$$E_{intra-task} = \sum P_{running} * T_{running\_i} + P_{idle} * T_{idle\_i} \tag{3.13}$$

where $P_{running}$ and P_{idle} are the power consumption of the idle and running states of a processor. $T_{running\_i}$ and T_{idle_i} are the times during which processor i is idle or running, respectively. Note that $T_{running\_i}$ is related to task execution times, which is considered in the STORM using the worst case (WCET). Table 3.4 gives the task temporal parameters used in the STORM environment for the H.264 video decoder application for the maximum frequency (720 MHz). The activation date of the last task is the same as the deadlines of the 4

previous tasks. This allows to liberate a processor for the last task. Thus, the design exploration targets architectures containing up to 6 processors.
In this case-study, we consider only one scheduling policy (the *SCHED_OTHER*) and one message size for Inter-Process Communication (8 KB). Therefore, the OS consumption calculated by the simulator depends only on the processor frequency. The simulator has to take into account the frequencies given by the DSR algorithm in order to estimate the consumption of both OS services and intra-task functions during simulation.

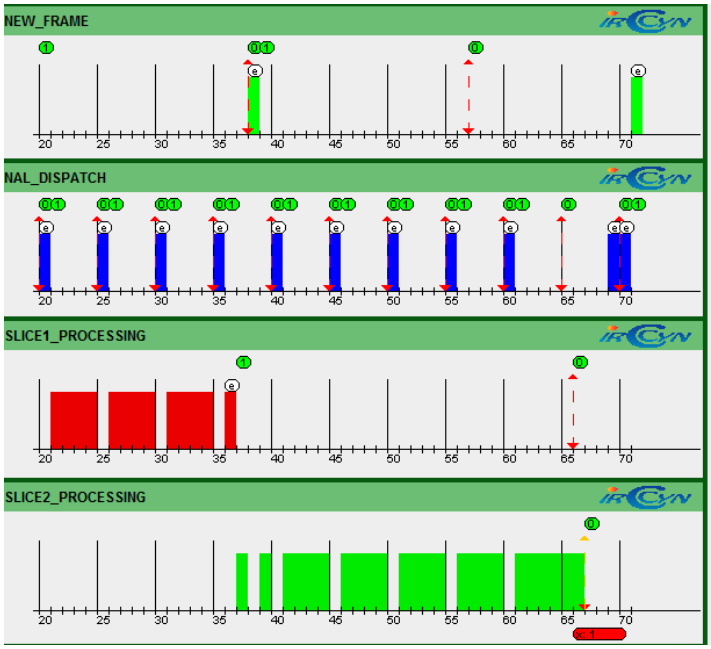

FIGURE 3.20 – Schedule of tasks using DSR technique for 1-processor configuration

Figure 3.20 gives an extract of the simulation trace for the one-processor configuration within the interval 20-70 ms. As shown in this figure, using only one processor leads to deadline violation. For example, the *SLICE2_PROCESSING* task is still not terminated at time instant 67 ms (its absolute deadline) and is interrupted by another task (*SLICE3_PROCESSING* not shown in the figure). The one-processor configuration will be thus excluded. Simulations showed that systems with more than one processor satisfied the required QoS of 15 frames per second. Since these systems satisfied the performance constraint, design space exploration will be based on the energy consumption results. Figure 3.21 gives the energy consumption results for the processing of one frame, when using the DSR technique for different initial frequencies. Processors are reinitialised using these frequencies before each frame processing. Figure 3.21 shows that over 4 processors, increasing the number of processors tends to be ineffective since it increases significantly energy consumption. Based on these results, we can consider systems with 2, 3 or 4 processors to be acceptable configurations. In order to decide, which configuration is the more suitable, more accurate energy estimation is needed. Therefore, these configurations will be further explored at the transactional level.
Table 3.5 gives the contribution of the OS services in the overall energy consumption when using the DSR technique with an initial frequency of 500 MHz. We note that the context switch is the most consuming service due to the high consumption of the frequency scaling

TABLE 3.5 – OS services energy consumption rates when using the DSR technique

Number of processors	Context switch	Inter-process communication	Scheduling routines
1	44%	35.4%	1.07%
2	35.38%	32.5%	1.12%
3	33.13%	29.6%	1.17%
4	30.88%	27.1%	1.38%
5	28.63%	23.8%	1.27%
6	26.8%	21.7%	1.8%

by the DSR technique. As shown in Table 3.5, when the number of processors increases the OS services consumption decreases. This is mainly due to the fact that the number of context switches from a task to another is reduced when the number of processors increases.

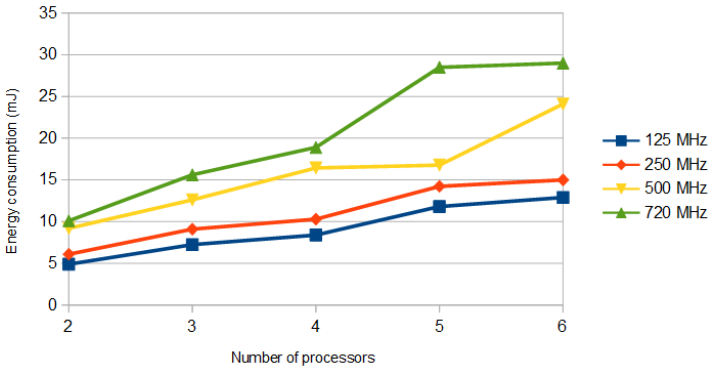

FIGURE 3.21 – Energy consumption (mJ) results given by the STORM simulator when using the DSR technique

3.7.2.3 Transactional-level energy/power exploration

As mentioned previously, the energy consumption estimation at the transactional level takes into account OS service and intra-task consumption in a more precise way by considering the micro-architectural activities. Transactional power modeling exploits the functional power model of frequency switching and gives a refined characterisation of the IPC, scheduling and intra-task consumption through the transactional power model presented in equation 3.11. In order to estimate the energy consumption of the application at the transactional level, energy models are integrated in the SoCLIB simulator. The DSR algorithm is also integrated in the simulator for a dynamic power optimization. The output frequencies of the DSR algorithm are considered by the simulator in order to estimate the frequency-switching consumption. These frequencies are used together with the Instruction Per Cycle and the

cache misses counters given by the simulator in order to estimate the consumptions related to micro-architectural activities.
The SoCLib simulator was extended to trace power consumption evolution during simulation in order to have more precise consumption results compared to the functional STORM simulator. Figure 3.22 gives an extract of the power variation of one processor in a 4-processor architecture, with an initial frequency of 720 MHz.
We note that, at the beginning of the execution, the power consumption (the processor frequency) does not change by the DSR. This is due to the fact that the processor is executing one of the first two tasks, where the difference between the actual and the worst execution time is insignificant. Starting the slice processing, we observe a tiling in terms of power consumption, which is due to a dynamic change of the operating frequency of the processor throughout the simulation.

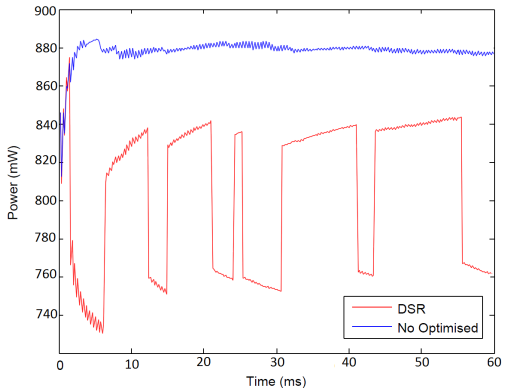

FIGURE 3.22 – Power variation of one processor during the simulation of a 4-processor architecture

Figure 3.23 gives a comparison between the functional and transactional energy consumption results. As shown in this figure, the error between the transactional and the functional levels can be significant (up to 75% for the 2-processor architecture). This can alter the exploration reliability of the functional level. Therefore, the rules to eliminate some configurations at the functional level should not be very strict. In this case study, even if the functional level indicates that the 2-processor architecture is the most suitable, we kept also the 3 and 4 processor configurations in order to have more accurate results through transactional-level exploration. The estimation at the transactional level is enough accurate to take more reliable decisions. In addition, this estimation will allow to calibrate the system battery and to perform an early thermal study of the final chip. The obtained transactional-level results showed that the 2-processor configuration is the most suitable for the considered H.264 application. However, targeting an ASIC realisation, the final choice of a given configuration (the number of processors to be implemented on the chip) will depend also on other parameters. First, an architecture is never designed to carry out only one application. The H.264 is chosen in our work as a reference presenting a large spectrum of video processing applications. Second, we have to consider future application updates that evolve rapidly in the embedded application domain. Therefore, the implemented architecture should offer the possibility to activate/deactivate processors according to application requirements and performance demand. In this context, modern chips such as the Tegra processor (quad-core) from NVIDIA can be used. Third and not least, thermal dissipation, the cost of the chip and

the size of the battery are also important parameters that can impact the final choice.

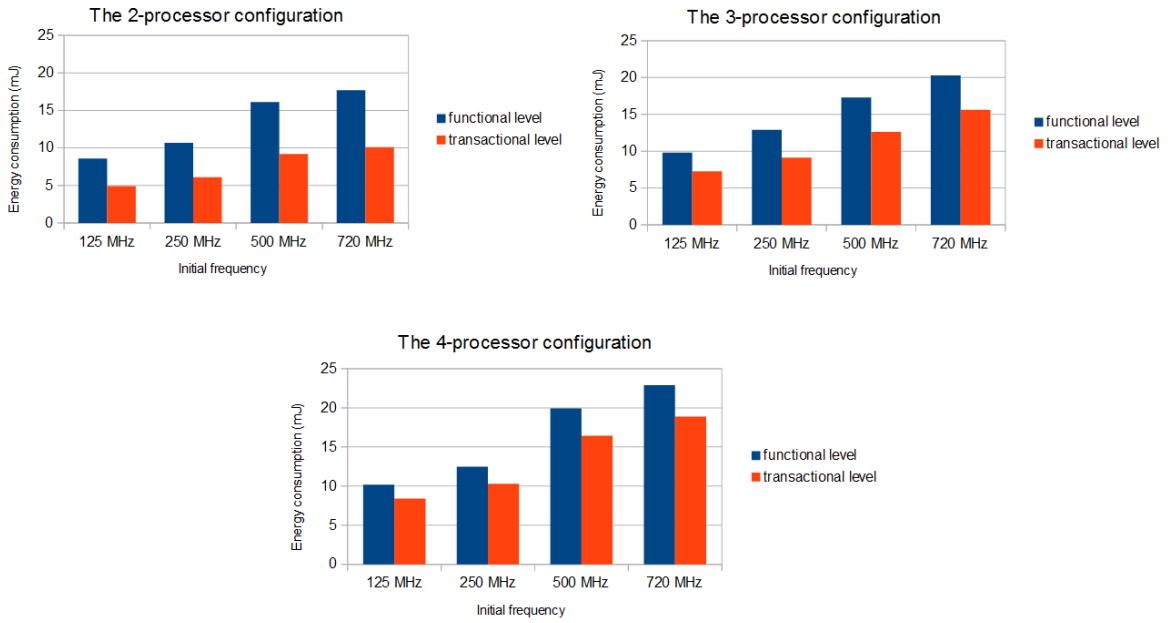

FIGURE 3.23 – Comparison between the functional and transactional energy consumption results

3.8 Model driven engineering :

MDE revolves around three focal concepts. Models, Metamodels and Model Transformations. A model is an abstract representation of some reality and has two key elements : concepts and relations. Concepts represent ”things” and relations are the ”links” between these things in reality. A model can be observed from different abstract point of views (views in MDE). The abstraction mechanism avoids dealing with details and eases re-usability. A metamodel is a collection of concepts and relations for describing a model using a model description language and defines syntax of a model. This relation is analogous to a text and its language grammar. Each model is said to conform to its metamodel at a higher definition level. Finally, MDE allows to separate the concerns in different models, allowing reuse of these models and keep them readable. The MDE development process starts from a high abstraction level and finishes at a targeted model, by flowing through intermediate levels of abstraction via Model Transformations (MTs) ; by which concrete results such as an executable model (or code) can be produced.

Gaspard2 [21] is an MDE oriented MPSoC co-design framework based on the Modeling and Analysis for Real Time and Embedded systems (MARTE) standard. The design flow in Gaspard2 follows several steps : system modeling and deployment, model transformations and code generation. The left side of figure 3.24 shows the design flow targeting the SystemC platform, which we used in our work. Gaspard2 targets also other platforms such as Fortran and VHDL. In Gaspard2, there is a separation between the architecture and the application models as shown in the left side of figure 3.24. An application model describes the

SW and/or HW components of an application. To link the application and the architecture models, an allocation model is used. At the deployment level, every elementary component (application or architecture component) of a system is linked to an existing code hence facilitating Intellectual Property (IP) reuse. Each elementary component can have several implementations (e.g., SystemC, VHDL). The deployment model provides IP information for model transformations targeting different domains (formal verification, simulations, high-performance computing or synthesis). After the deployment phase, model transformations (transformation chains) permit to add some details to the input model in order to get closer to the targeted technologies. At the end of a transformation chain, we have a model with technical details allowing the code generation related to the targeted technology.

Our contribution in Gaspard2 was to integrate the power estimation concepts in this framework. The integration follows two steps as shown in the right side of figure 3.24. The first step is to generate power estimation modules for the SystemC IPs in the Gaspard2 IP library. For this, we developed a new profile and a new metamodel dedicated to power estimation. The profile allows to describe the behavior of the power estimators and to determine the architectural parameters that will be used in the consumption estimation later, when the estimators will be integrated into a whole MPSoC architecture. Using model transformations, the SystemC code of the estimators is generated automatically, in order to obtain new IPs dedicated to power estimation. These IPs are integrated in the IP library of Gaspard2. They are ready to be used for the consumption estimation of the hardware components used in the Gaspard2 system designs. The second step is to integrate the power estimation in MPSoC models in Gaspard2 in order to automate the power estimation for these systems. For this, we extended the deployment profile of Gaspard2. This extension allows to link the existing estimators in the library to the hardware components of a modeled MPSoC. The MDE approach of Gaspard2 allows then to generate the SystemC code of the whole system with the integrated estimators. This code can be then integrated into the SystemC simulator and the energy estimates can be displayed during the simulation. All these contributions are detailed in [39].

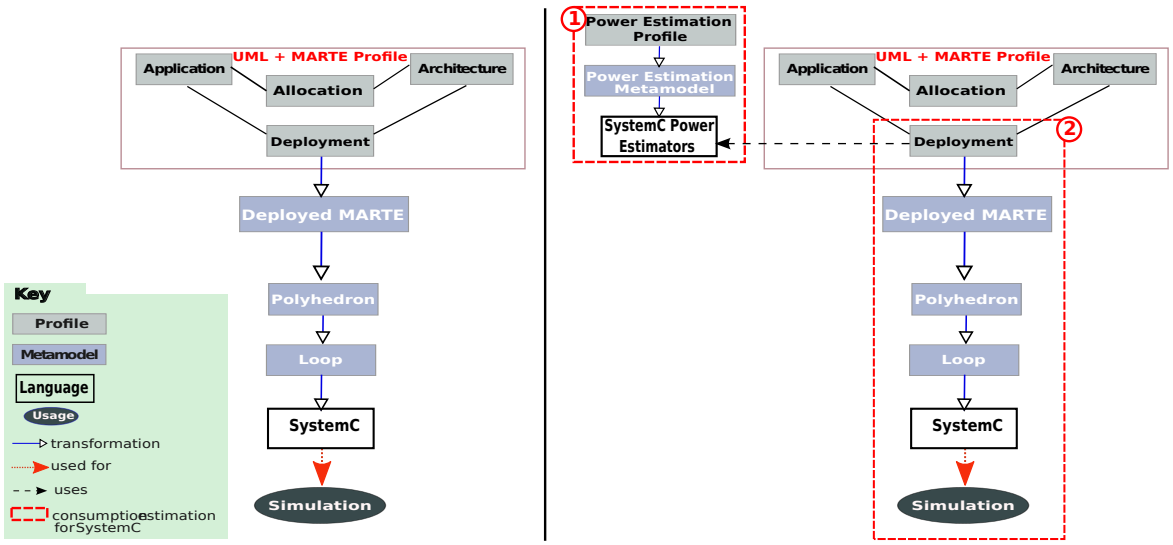

FIGURE 3.24 – Power estimation integration in the Gaspard2 framework

3.9 Discussion

Summary. In this chapter, we summarized our contributions in the field of low power design targeting homogeneous and heterogeneous MPSoC. The challenge lay in raising the issues related to the design methodology, the power modeling, the complex design space exploration, and the software development environment. We tackled these issues that are gripping embedded system designers by means of proposing an efficient multilevel power-aware design methodology. The first attractive aspect in our proposal is the global vision while considering the power consumption of the system. Indeed, we addressed the global system consumption that includes processors, memories, reconfigurable circuits, operating system services, etc. The second attractive aspect in the proposed methodology is the power modeling approach used. Actually, FLPA was used to develop power models for hardware and software components well-adapted to the system level design. these power models are integrated in the design flow in the context of multi-level design space exploration : to refine power and energy estimations, they are used in conjunction with simulation tools at different abstraction levels. The main advantage of such approach is to abstract the conventional electronic details linked to power estimation essentially for software designers. The integration of runtime power management techniques in the design flow is another benefit of our methodology. As a cost-effective software development approach, we relied on MDE to abstract the concepts and to automate the design process of MPSoC while considering the power metric.

Limitations. The presented energy/power-aware design methodology can have a strong impact on the adequate chip specification for a given application domain. In addition, the integration of runtime optimisation techniques such as DVFS can lead to a better energy saving. However, we think that this will not be sufficient for next sub-micron technologies, especially with the dark silicon issue where the system should be configured at runtime to exploit the strict necessary resources. We should include hardware and software components for runtime power monitoring that should be managed by the execution model. At the programming level, there are also requirements in terms of language expressiveness to take into consideration the dynamicity of the hardware for better energy efficiency.

The future. In the future, we will pursue this research direction considering the fact that IC 2D scaling is reaching the fundamental limits. Hence, the semiconductor actors are exploring the use of the vertical dimension (3D) for logic and memory devices. The combination of 3D device and low power device will introduce a new era of scaling, identified in short as *3D Power Scaling* [83]. Furthermore, thermal effects are exacerbated in 3D technology. So, we need to rethink about the power management for the next generation of 3D-based multiprocessor SoC or 3D FPGA.

Summary	Supervision :	- **Santhosh Kumar Rethinagiri**, PhD in computer science, graduated in March 2013, now senior researcher at Barcelona Supercomputing Center. - **Imen Mhedbi**, Master degree (March - September 2011), now third year PhD student at the University of Pierre et Marie Curie, LIP6 Laboratory. - **Yomna Ben Jmaa**, Master degree (April - September 2012), now first year PhD student at the University of Valenciennes, LAMIH Laboratory.
	Journals :	IEEE Transactions on Industrial Informatics [16], EURASIP Journal of Embedded Systems [39], ACM Transactions on Embedded Computing Systems [21], IEEE Embedded Systems Letters [1]
	Selected conferences :	IEEE ICCD 2011 [29] (Best paper awards), ACM GLSVLSI 2012 [28], ISQED 2014 [30]
	Realization :	PETS tool [30]
	Funding :	ANR OPEN-PEOPLE, INRIA internships program

Chapter 4

Massively Parallel Dynamically Reconfigurable Execution Model

This chapter presents my contributions since 2011 at INRIA DreamPal team in the field of massively parallel dynamically reconfigurable execution model. Mainly, these works cover the PhD thesis of Venkatasubramanian Viswanathan started in February 2012 and will be defended in January 2015 (co-advised with Jean-Luc Dekeyser). We are pursuing investigation for this field in the frame of the PhD thesis of Karim Mohamed Ali and Wissem Chouchene who started in October 2013.

The rest of the chapter is organized as follows : in Section 4.1, we introduce the main challenges of the design of parallel and dynamic reconfigurable architectures ; Section 4.2 gives a summary of the related works. Section 4.3 details the design of a multi-FPGA System-in-Package that supports massively parallel dynamically reconfigurable execution model. Section 4.4 presents the parallel reconfiguration model for 3D FPGA next generation. To evaluate our approach, experimental results are presented in Section 4.5. In Section 4.6, we introduce the HoMade processor featuring massively parallel dynamically reconfigurable execution model. Finally, Section 4.7 discusses the strengths, limitations and future directions to the presented works.

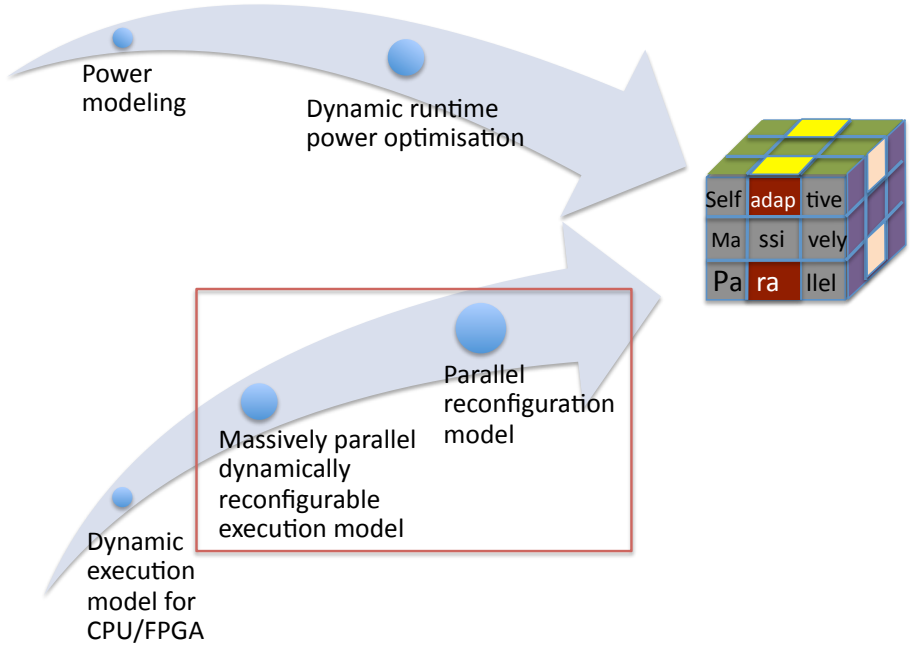

FIGURE 4.1 – Contributions the field of parallel and dynamic execution model

4.1 Overview of main challenges

Standard Integrated Circuits (IC) are reaching their limits and need to evolve in order to meet the requirements of next generation computing. One of the most promising evolutions is the 3D-Stacked Integrated Circuits (3D SICs) [138]. An excerpt from the HiPEAC vision [83] promotes that *"The advent of 3D stacking enables higher levels of integration and reduced costs for off-chip communications. The overall complexity is managed due to the separation in different dies, independently designed"*.

In parallel, FPGAs (Field Programmable Gate Arrays) have emerged as a privileged target platform for intensive signal processing applications [136]. Indeed, FPGAs have the benefits of high speed and adaptability to the application constraints with a reduced performance per watt comparing to the General Purpose Processors (GPP). They offer inexpensive and fast programmable hardware on some of the most advanced fabrication processes.

We believe that 3D integration will lead to a significant shift in the design of FPGA circuits. 3D integration will vastly increase the integration capabilities of FPGA circuits. Specially, the monolithic 3D FPGAs promote a very important benefit over 2D FPGAs [105]. Due to their structure, monolithic FPGA presents a 3D architectural transformation from a 2D FPGA to offer higher performance. Indeed, as shown in figure 4.2 (a), the 2D FPGA consists of a reproduction of a basic building blocks called "Tile" interconnected through regular queues. Each tile is composed of three types of resources, connection box (CB), logic block (LB) and switch box (SB). The objective of the architectural transformation is to reduce the horizontal wired connections to avoid the synchronization errors and to optimise energy consumption and area. Moving from 2D to 3D FPGAs consists in organising each type of resources cited above in a silicon layer and superposing them one upon the other in the same chip using interconnection TSV (Through-Silicon-Vias) [105].

The stacking order plays an important role in ensuring good performance of 3D FPGAs according to [106] [46] [132]. Figure 4.2 (b) illustrates the general structure of a 3D layered FPGA. Indeed, the memory layer is located above the other layers ; it is built with memory SRAM blocks of two types : LB-SRAM (Logic Block SRAM) and RR-SRAM (Routing Resource SRAM). The middle layer is the routing layer which is a mesh of nodes, where each node is composed of four CBs and four SBs. Finally, the lower layer composed of LBs interconnected uniformly.

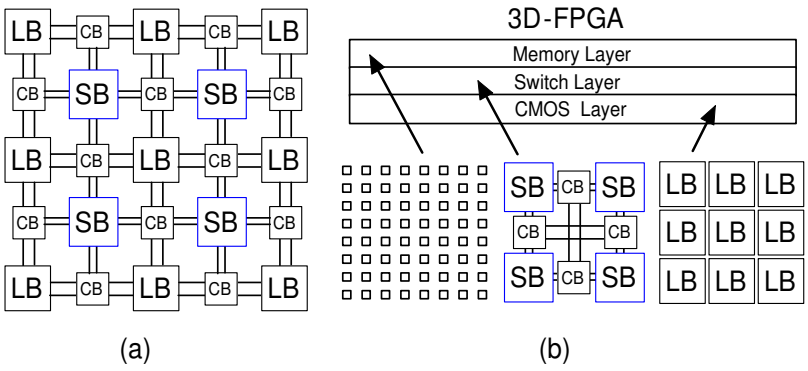

FIGURE 4.2 – (a) 2D-FPGA (LB : logic block, CB : connection box, SB : switch box). (b) 3D Monolithically stacked 3D-FPGA [105].

The convergence of massive parallelism and dynamic reconfiguration is inevitable : we believe it is one of the main challenges in computing for the current decade. By incorporating the configuration and/or data/program memory on the top of the FPGA fabric, with fast and numerous connections between memory and elementary logic blocks, it will be possible to obtain dynamically reconfigurable computing platforms with a very high reconfiguration rate [131]. Such a high dynamic reconfiguration rate was not possible before, due to the serial nature of the interface between the configuration memory and the FPGA fabric itself. We highlight that due to thermal issues, not all TSV interconnections can be used in a 3D FPGA.

It was demonstrated that less than 10% of these interconnections can be used depending on the number of layers and the sub-micron technology [93] [73].

The FPGA technology also enables massively parallel architectures due to the large number of programmable logic fabrics available on the chip. Single Program Multiple Data (SPMD) is the favourite execution model for designers while implementing parallel architectures on FPGA. For instance, Xilinx demonstrated 3600 8-bit picoBlaze softcore processors running simultaneously on the Virtex7 2000T FPGA. For specific applications, picoBlaze can be replaced by specialized hardware accelerators or other IPs (Intellectual Property) components. This opens the possibility of creating massively parallel IP-based machines. Such architectures can be customized at runtime using the Dynamic Partial Reconfiguration (DPR) feature, a reconfiguration that can be done in parallel for all or for a subset of the IPs.

Recently, SIC technology, also known as 2.5D ICs, has been released by the manufacturer Xilinx for the Virtex 7 FPGA family. Such technology is considered as a near-cousin to 3D. The next generation 3D FPGAs will allow efficient dynamic reconfigurations in a massively parallel manner. The 3D FPGA realised in [72] confirms this analysis. As shown in figure 4.3, several FPGA tiles can be configured in parallel while accessing to different configuration memory banks simultaneously. ***Hence, we can obtain a reconfiguration model well-traced on the execution model (SPMD) in order to reconfigure a subset of the parallel computing nodes***. Software applications running on such hardware can then efficiently reconfigure the hardware at runtime according to their needs, thereby achieving significant savings in circuit space, energy consumption, and execution time [82]. We anticipate that FPGAs will play a major role in 3D evolution : FPGAs are currently one or two generations behind the most advanced technologies for standard processors, but their application as specific hardware is an order of magnitude faster than software solutions on standard processors [54]. One of the most promising evolutions are next generation 3D FPGAs, which will enable users to build dynamically reconfigurable massively parallel hardware architectures around them. This new paradigm opens many opportunities for research, since, to our best knowledge, since there are no parallel reconfiguration models for such technology, no execution models for massively parallel and dynamically reconfigurable architectures on 3D FPGA, and no dedicated tools for mapping those architectures on 3D FPGA or estimating their performances.

We shall address the following topics : proposing an execution model and a design environment, in which users can build customized massively parallel dynamically reconfigurable hardware architectures, benefiting from the reconfiguration speed an parallelism of 3D FPGAs ; proposing dedicated language for programming applications on such architectures and designing the appropriate prototyping environment.

Defining execution model for the new architectures : The execution model for the new architectures has to incorporate massive parallelism and dynamicity in order to benefit from the very large size of the next-generation FPGAs, which currently contain several million of logic gates. Furthermore, we have to offer the flexibility for designers to choose the granularity of their IPs. Hence the execution model can implement fine-grained as well as coarse-grained computation to address the heterogeneity issue. ***In other words, parallelism, dynamicity, and heterogeneity should be addressed at the same level while programming reconfigurable architectures***.

The first aspect (dynamicity) consists in designing a processor with only few hard-coded instructions for controlling the execution flow. All the other instructions reside in dynamic IP units that are instantiated, executed (possibly in parallel), and replaced at runtime. The IPs can be arbitrary complex and heterogeneous, including ALU functions, load/store units,

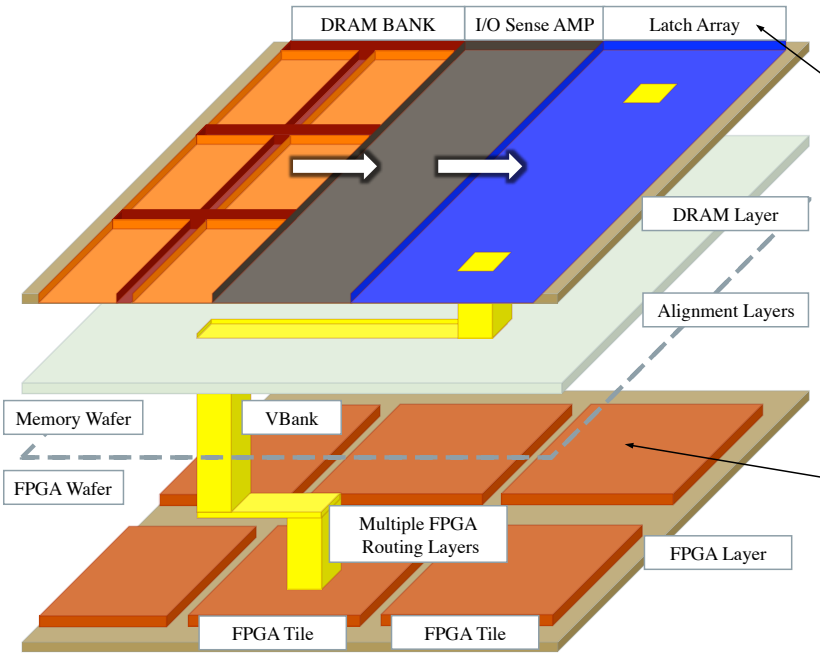

FIGURE 4.3 – Example of 3D FPGA realisation [72]

or even other processors. Users can pre-select a set of IPs to be implemented on FPGA according to the needs of their application. By taking advantage of the dynamic reconfiguration functionality of new FPGAs, the set of IPs associated to a given processor can evolve during the execution of the program : some are instantiated, others are removed, etc. In the second aspect of the defined execution model (parallelism), the processing element (PE) of the FPGA circuits will have to be completed with specific hardware to manage the massively parallel processing (network, synchronization barriers, communication...), and we envisage, in the continuity of our team research on mppSoC [108] [99], that the execution model of the whole system will be an extension of the old SPMD model (Single Program Multiple Data, i.e., executing the same code in parallel on many PEs).

Defining a parallel reconfiguration model for 3D FPGA : The emerging technology of programmable logic devices provides denser logic to accommodate massively parallel architectures. However, the reconfiguration models of these architectures still remain sequential. Basing on the existing 3D technology, we have to study the global structure of the FPGA and the disposal of the logic layers in relation to the reconfiguration memory layers. High-speed reconfiguration controller architecture should be defined according to the FPGA structure. Within this structure, we will consider coarse-grain reconfiguration granularity. We should be able to map an IP on several reconfigurable regions (for SPMD execution model) without a pre-allocation process. The academic merit of such an approach can be served as a new idea for hardware virtualization.

Designing a Multi-FPGA SiP for massively parallel dynamically reconfigurable architecture : With the aid of FPGA-based prototyping tools, it is possible to have the proposed execution model running on a multi FPGA platform. With this approach it is not mandatory to develop the physical layout of the 3D FPGA, which is out of the scope of our research field. It is also feasible to make some parts of the execution model running on hardware and

others defined as a virtual prototype. This does not only make the development of the 3D architecture faster and more flexible, but also facilitate the testing phase. In addition, the use of a multi FPGA platform may become a necessity due to the complexity and the size of the FPGA to be developed. Without such a tool, it is not efficient to develop the model using the traditional computing platforms. This is because of the heavy computational tasks required to represent it (e.g. simulating the FPGA resources, execution of the mapping and routing algorithms), or to be executed on it (e.g. having a soft microprocessor in the developed 3D FPGA on which a reconfiguration algorithm runs). Although the emulation will be restricted to a limited number of FPGAs, a proof of concept will be extended to N number of FPGAs.

4.2 Related works

One of the Difficult Challenges through 2019 identified by ITRS in its 2012 report [29] is : *Providing a capability for more rapid adaptation, reuse and reconfiguration*. This challenge has attracted several national, European and international projects related to IP reuse, reconfiguration, parallel systems and verification in systems design. However, to our best knowledge, none of the existing projects proposes dynamic reconfigurable parallel architectures and languages designed with (and connected to) formal tools for safely programming applications on these architectures.

In the literature, several execution models have been proposed to deal with the parallelism, heterogeneity, and the dynamicity dimensions in embedded architectures. As a first alternative, we quote the ASIP (Application Specific Instruction-set Processors) approach [95], which allow some customization in the hardware : instruction set extensions, instruction parametrization and inclusion/exclusion of predefined components (IPs) for specific applications. For example the eMIPS (extensible MIPS) [122] is an extensible processor architecture that achieves the performance of application-specific hardware optimizations in a safe and general-purpose environment. It allows multiple secure extensions to load dynamically and to plug into the stages of a pipelined data path, thereby extending the core instruction set of the microprocessor. Extensions can also be used to realize on-chip peripherals and, if the surface permits it, even multiple cores [75]. However, with the ASIC static customization a large percent of the CPU core is static too, and consumes resources (surface and power) even when it is not used. This consumption can become very expensive in a massively parallel integration scenario, where the resource waste would be multiplied by the number of elementary nodes. Regarding parallelism, even if some ASIC solutions integrate multicores, they remain far from massively parallel processing.

There is also a second alternative to converge towards dynamic parallel embedded systems. With the advance of the system-level integration, the high-tech industry is now moving toward multiple-core parallel architectures build around networks on chip. ADRES (architecture for dynamically reconfigurable embedded systems) is a flexible, high-performance architecture template for low-power embedded applications. It consists of a tightly coupled Very Long Instruction Word (VLIW) processor and a coarse-grained reconfigurable memory array. ADRES is a flexible template to generate concrete instances. Together with a simulator and an xANSI C compiler, this tool chain allows for architecture exploration and development of application-domain-specific processors [66]. However, the customization allowed

29. http ://www.itrs.net/Links/2012ITRS/Home2012.htm

by this framework is static (done before execution time), thus, all components consume resources independently of whether they are used or not at a given time.

P2012 [64] is an embedded computing platform based on multiple globally asynchronous, locally synchronous (GALS) clusters featuring up to 16 processors. Each processor can be customized at design time with modular extensions (vector units, floating point units, special purpose instructions). Clusters can easily become heterogeneous computing engines thanks to the integration of coarse-grained hardware accelerators. The execution model of P2012 is mainly MPMD using standard OpenCL and OpenMP parallel codes as well as proprietary Native Programming Model (NPM). This project does not deal with dynamicity for tuning the design according the application's behaviour. Heterogeneous clusters make the software development of such systems very difficult.

RAMPSoC stands for Runtime Adaptive MultiProcessor SystemonChip [94]. It is a heterogeneous and runtime adaptable platform. This project focuses on improving MPSoCs by allowing the runtime reconfiguration of heterogeneous processor cores, their communication infrastructure and special accelerator units. In addition to processor cores, special Finite State Machines (FSMs) can be configured in RAMPSoC, allowing a flexible adaption of the hardware. This project does not take into account the regularity of MPSoC. Their model implies a large space occupation for a full core and limits the maximum number of available cores/FSMs.

In the FP7 European project REFLECT [30] (Rendering FPGAs to MultiCore Embedded Computing), the partners addressed the challenges of developing, implementing and evaluating a compilation and synthesis system approach for FPGA-based platforms. They intended to solve some of the problems that arise when mapping efficiently computations to FPGA-based systems. In particular, the use of aspects and strategies will allow developers to try different design patterns and to achieve design solutions guided by non-functional requirements. This design flow seems a good compilation chain taking into account non functional properties, however it does not address parallel architectures.

The FP7 European project HEAP [31] (A Highly Efficient Adaptive multi Processor-framework), dealt with an interesting problem in the development process for multicore and multi-threaded architectures : the identification of a sufficient amount of thread level parallelism to exploit the available hardware. It facilitates parallelization of the code to fit on existing multi core architectures. This project focuses only on C/C++ code parallelization at a high level without the possibility of using customised HW architectures.

At the technological level, the FlexTiles [32] project aims at designing 3D stacked chips with a manycore layer and a reconfigurable layer. A virtualisation layer on top of a kernel hides the heterogeneity and provides self-adaptation capabilities by dynamically relocation of application tasks to software on the manycore or to hardware on the reconfigurable area. The reconfigurable technology is based on a virtual bitstream that allows dynamic relocation of accelerators just as software based on virtual binary code allows task relocation. This project targets complex and heterogeneous architectures, which makes it unsuitable for regular applications. It also uses a kind of predefined hardware architecture that cannot be changed. In [72] proposes the use of 3D integration technology to enable low overhead reconfigurable computing. To reconfigure an FPGA, a configuration is read from the DRAM into a latch

30. http ://www.reflect-project.eu/
31. http ://www.fp7-heap.eu/
32. http ://flextiles.eu/WordPress3/

array at runtime ; then, the configuration is loaded from the latch array into the FPGA in 5 cycles (60ns). They demonstrate that 3D configuration caching works best when used in conjunction with FPGA-based accelerators, rather than pure FPGA-based systems ; in these systems, the reconfiguration latency can easily be hidden behind software execution on the processor controlling the accelerator. This recent work confirms the fact that 3D integration will enable an efficient reconfiguration on the future 3D FPGAs.

Our proposal is to do most of the reconfiguration dynamically, so that a component that becomes inactive can be removed from the system and the resources it consumed can be reassigned to active components ; and to take advantage from the massive parallelism of the new 3D FPGA.

4.3 The Multi FPGA System-in-Package

Targeting adaptive and intensive signal processing applications that can make profit from a massively parallel dynamically reconfigurable execution model, we specified an appropriate platform for such application domain. The problem in today's technology exists at two levels, technological level and architectural level. At the architectural level, the industrial and academic state-of-the-art shows a lack of embedded systems supporting massively parallel dynamically reconfigurable execution model to satisfy the application requirements. At the technological level, there is a lack of customizable reconfigurable computing power that can be changed according to the evolving needs of the application. Since these needs evolve over time, the underlying hardware itself has to change according to the requirements of the application. Hence, the hardware used for a specific application is subject to obsolescence. Redesign of the entire systems results in huge Non Recurring Engineering (NRE) costs and increased time to market. Therefore there is a strong need for a customizable obsolescence proof reconfigurable computing technology.

The use of a customizable and runtime reconfigurable hardware can remove the obsolescence issues to the computing technology used in our target application domain. Using multiple FPGAs at the same time can provide computing power comparable to that of ten CPUs while guaranteeing real-time operation. Another essential attraction of reconfigurable components is the easy management of the degree of parallelism allowing us to control the execution time of tasks based on the needs of the application.

As shown in figure 4.4, the prototype of the board is realized with a carrier board and 4 FPGA modules. FPGA modules contain only the FPGA and the need electrical circuitry to support the operation of the FPGA. The connector on the FPGA module is used to mate with the carrier board. The FPGA module size is always fixed to 125x95 mm. The size of the FPGA module has been chosen taking into consideration the size of the largest FPGA device in the market. The FPGA modules are connected via a PCIe switch. One of the key features of this board is the customizable and scalable communication link for high-bandwidth low-latency peer-to-peer communication. Communication between several nodes can happen in parallel without any latency in between. On the other hand, the carrier board consists of all the other components and features (i.e., FMC, memory, PCIe switch, COM express and peripheral I/O interfaces) of the multi-FPGA board. The other components such as FMC, memory, COM Express, peripheral I/Os used and the connections between them are illustrated in figure 4.4.

The key architectural feature of this board is to be able to reconfigure more than one node at the same time. Using the PCIe switch, the configuration data can be multicast/broadcast

to more than one node at the same time. The configuration data will be stored in the local memory of the respective nodes. Later reconfiguration of multiple nodes can be initiated by a broadcasting a reconfiguration command.

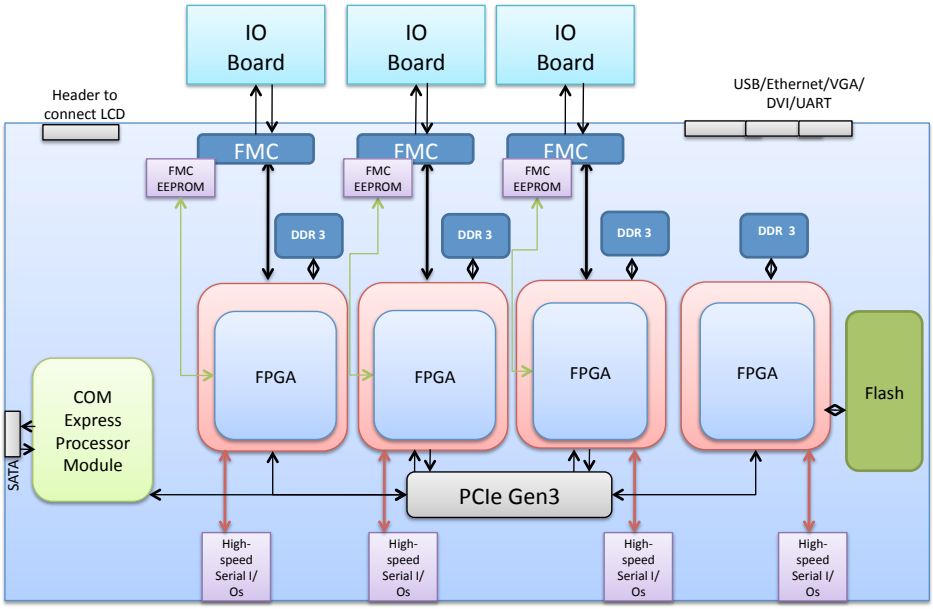

FIGURE 4.4 – Architecture of the multi-FPGA board

Parallel I/O management model

Targeting parallel architecture, designers have to manage the I/O data efficiently, otherwise the system loses its attraction in terms of performance. In our board, we will rely on the FPGA Mezzanine Card (FMC) standard proposed by the VMEbus International Trade Association (VITA) group [142] to solve the I/O obsolescence issue as discussed in Chapter 2. As shown in figure 4.5, each FPGA can communicate with the outer world through FMC module. In case of distributed processing, the data received via a single FMC might need to be shared with more than one FPGA. In this case, the owner of FMC I/O can share its data via the PCIe switch. We can also reconfigure our board as a parallel reconfigurable machine with a shared memory model. Each FPGA has a local DDR3 memory while the master FPGA has a memory size four times compared to the local one. Each FPGA can store its complete local memory in the global shared memory via the PCIe switch. The Master FPGA in turn can retransmit the data to one or more nodes at the same time if requested thus forming the notion of a global shared memory.

All the communication between the FPGA modules takes place via the PCIe Switch on the carrier board. There are two important aspects of communication. First, there are the architectural requirements for communication i.e. the different communication models are required. Second is the configuration of the switch for each communication model. A device is capable of communication with one or more devices at the same time. Therefore, all the communication falls into two broad categories :

– One-to-one communication (Data sharing and I/O sharing between devices)
– One- to-many communication (Configuration data transfer to several devices)

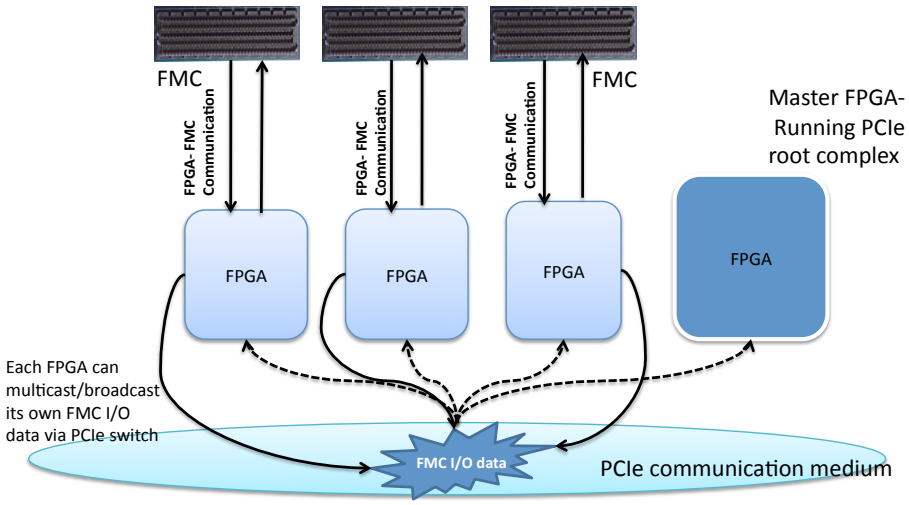

FIGURE 4.5 – Parallel I/O model using FMCs and PCIe Switch

All types of communications are handled by the PCIe switch as Transaction Level Packets (TLPs). There are different types of TLPs (A, B, and C) and any communication will have to fall into these types of transaction. While all one-to-one transactions are handled by one of the PCIe TLP type, one-to-many communication is handled by multicast protocol provided by the switch.

4.3.1 The prototyping environment

The proposed system consists of a general purpose processor (intel, AMD, etc.) coupled with a multi-FPGA board with a PCIe x16 slot. The multi-FPGA board consists of four detachable FPGA modules and a PCIe switch on the back-plane, for communication between the FPGA modules and the host. Since the technology used in our multi-FPGA board is patent pending, we have used a multi-FPGA board from Pico Computing [42] which will be sufficient to demonstrate the main idea of parallel reconfiguration model, well-traced on the SPMD execution model.

The multi-FPGA board has a module-based FPGA computing power, that allows to add or remove any number of FPGA modules. Thus, system designers will be able to augment or remove hardware computing power as needed. This feature, combined with a scalable PCIe based peer-to-peer communication protocol, allows to practically add any number of computing nodes in the system. This setup with a PCIe switch truly enables us to emulate a 3D FPGA based architecture, where each FPGA can at some level be compared to a layer of the SIC, and be programmed individually to host a separate layer of configurable logic. Moreover, the PCIe switch in the system also provides one-to-many and one-to-all communication enabling a fully connected communication topology. This topology can be used to reconfigure several nodes or a subset of nodes in parallel, by broadcasting bitstreams to different FPGAs. Thus, it emulates the switch layer of the 3D FPGAs that allows to communicate, to broadcast, and to reconfigure several SIC layers in parallel, following the circuit structure of figure 4.2 and figure 4.3.

The communication between FPGAs and the host is based on Gen2 PCIe switch, which is a fully switched architecture. This means that multiple FPGAs along with the host processor

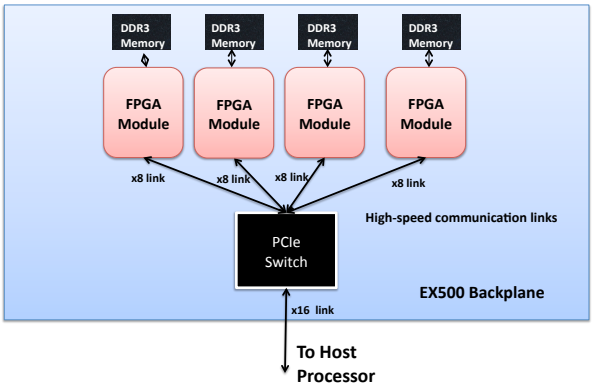

FIGURE 4.6 – Communication between host and FPGAs

can send and receive data simultaneously over the switch. Each FPGA module connects with each other via the PCIe switch with a x8 lane width, while the host processor connects to the PCIe switch using a x16 lane width as shown in figure 4.6. The PCIe tree is used to connect multiple peripherals, where the FPGAs are the leaves and the host CPU along with the I/O Hub, is the root. The host processor runs an Ubuntu Linux distribution with the drivers and software APIs that interacts with all the FPGA modules on the board. Furthermore, the driver supports the usage of POSIX threads. This allows the user to interact with multiple devices or even access multiple DMA channels on the same device concurrently. Each FPGA module also has a 4 GB of local DDR3 memory to store the bitstreams to reconfigure each FPGA at runtime, and also to share these bitstreams with other FPGAs. Several features of the architecture are explained in the following subsections.

4.3.1.1 Communication

The communication between the host to FPGAs, FPGA to FPGA and FPGA to the host is categorized into two distinct types i.e., bus-based communication and stream-based communication. Bus-based transactions provide register access on the PCIe address space, while stream-based transactions provide continuous burst transfers between two end-points or the host and end-point. As the names suggest, the former is used for command and control purposes, while the latter is used for high-throughput data transfers.

Bus-based communication

The bus model connects the software and firmware by presenting the firmware as a slave on a memory-mapped bus, which is driven by the software running on the host. In other words, it is a memory-mapped paradigm used by host software for both writing data to the firmware, and reading data from the firmware. Each transaction involves both data and an address and it models a set of registers. We use this type of transaction to send and receive command and status, to synchronize the start of a parallel reconfiguration and to monitor the process.

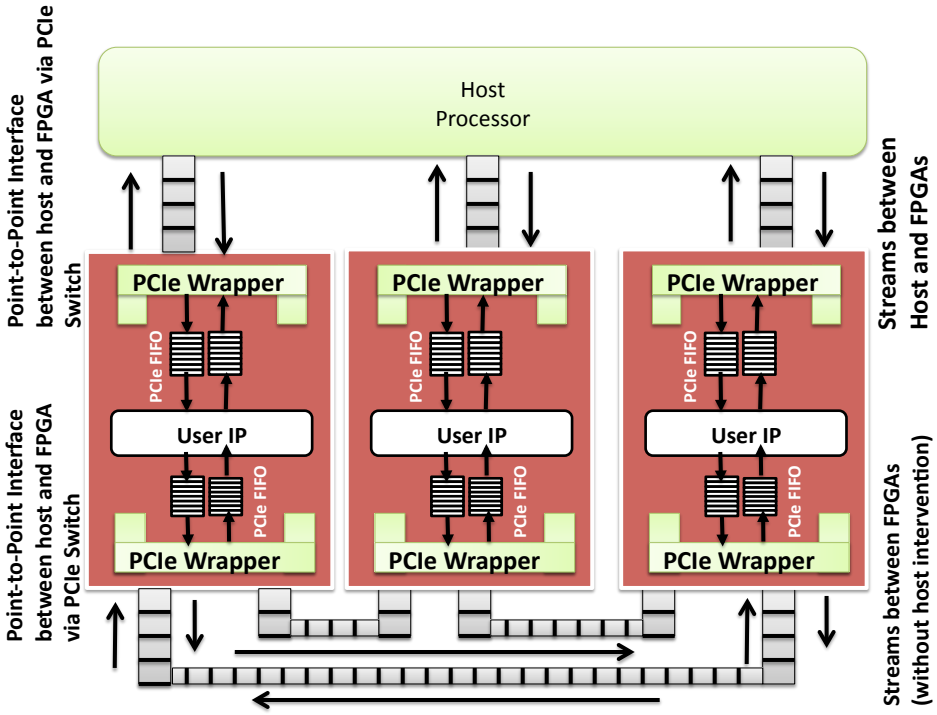

FIGURE 4.7 – A flexible communication topology using PCIe Streams

Stream-based communication

The stream method of communication is used for DMA transfers between the host memory and FPGA or between two FPGAs without processor intervention. It is simply a PCIe based DMA that serves as inputs to FPGAs or outputs from FPGA. A typical system will have at least one stream in each direction for bidirectional communication. A stream is a sequence of data, with flow control without addresses, implemented using a PCIe AXI FIFO. Streams are the fastest and most efficient way of moving data to and from FPGAs, since data is simply sent and received in-order. The software API functions can be used to know when the stream is available for reading or writing, by checking the number of available bytes in each stream. Each node can handle up to a maximum of 32 full-duplex streams. Figure 4.7 shows a flexible communication topology between the host and multiple FPGAs using Streams. We make profit from this flexible communication topology, to broadcast partial bitstreams from the host to multiple FPGAs, from FPGA to multiple FPGAs, at the same time via the PCIe switch.

4.3.1.2 Parallel Execution Model using POSIX Threads

The architecture also provides support for a parallel execution model, in addition to normal sequential execution of an application. In a SPMD execution model on a 3D architecture, same instance of an IP can be duplicated several times on different layers of the SIC. Thus it will be necessary for the software application to read/write and synchronize the software accesses to all these processes in parallel. In such cases using multiple threads, one writing to a process, another for reading the results and passing it on to another process, while the first

process is still writing more input, will be very useful. In order to facilitate this, the system allows the creation of multiple threads using POSIX library (PThreads). However, the user has to make sure that all the race conditions are handled. For example, calls to access the FPGA from multiple threads on different streams are always thread-safe. Moreover, input stream and an output stream that share the same number are also thread-safe. However, two threads writing to the same stream, or two threads reading from the same stream need to be thread-safe with mutual exclusions. Therefore, in our system, using PThreads in conjunction with streams, we can broadcast data to all the devices in parallel, thus forming the basis of a parallel reconfiguration model to emulate 3D FPGAs using several 2D FPGAs and a PCIe switch.

4.4 Parallel reconfiguration model

Using the features described in the previous sections, we aim to emulate a parallel reconfiguration model of 3D FPGAs. In a SPMD architecture, where multiple instances of the same IP process different data sets, we should reconfigure several IPs or a subset of IPs when the context of the application changes. In practical terms, it will not be efficient when reconfiguration is done sequentially for a large number of IPs, using current generation 2D FPGA based reconfiguration model. Thus we speculate that 3D FPGAs should and will be able to provide advanced reconfiguration mechanisms, where SPMD applications running on such high density devices, will be able to reconfigure several layers of SIC in parallel. In this context, we present a very common industrial application based on the SPMD execution model, that will be an ideal candidate to emulate the parallel reconfiguration feature of the next generation 3D FPGAs.

4.4.1 Modes of Parallel Dynamic Reconfiguration

This section discusses the ways in which a parallel reconfiguration can be initiated in our system. There are two ways for doing this. First, parallel reconfiguration can be initiated by sending the bitstream from the host processor to each FPGA in parallel via PCIe DMA streams as shown in figure 4.6. This is followed by a broadcast of a start command to all the FPGAs using the PCIe registers. The second option is to store the bitstream in each FPGA local DDR3 individually and then broadcast a reconfiguration start command to all the FPGAs at the same time, so that each FPGA can start reading from the respective local memory and start the process of reconfiguration. Broadcasting and communicating with more than one FPGA device at the same time has already been explained in Section 4.3 under PThreads. Also, the ICAP (Internal Configuration Access Port) controller that we have designed is interfaced to both PCIe DMA stream and the DDR3 memory, so that it can read the partial bitstream from both the host as well as the local DDR3 memory.

4.4.2 ICAP controller

The ICAP controller is designed as a simple state machine as shown in figure 4.8. Once the start command is received by the ICAP controller, it waits for a valid data either from the DMA stream from the host to FPGA or from the DDR3 memory stream depending on the type of the start command received by the controller. Once valid data is detected from

either of the streams, the ICAP controller starts and the data flow is continuous without any interruptions. The configuration data of the bitstreams follow the Xilinx .bin format starting with a bus width detection word, a sync word, followed by the configuration words. However, since the size of the bitstream is not encoded in the Xilinx .bin format, it is necessary to inform the ICAP controller about the size of the bistream along with start command using the PCIe registers. Once the controller finishes reading the bitstream of specified length, it notifies the host through status registers.

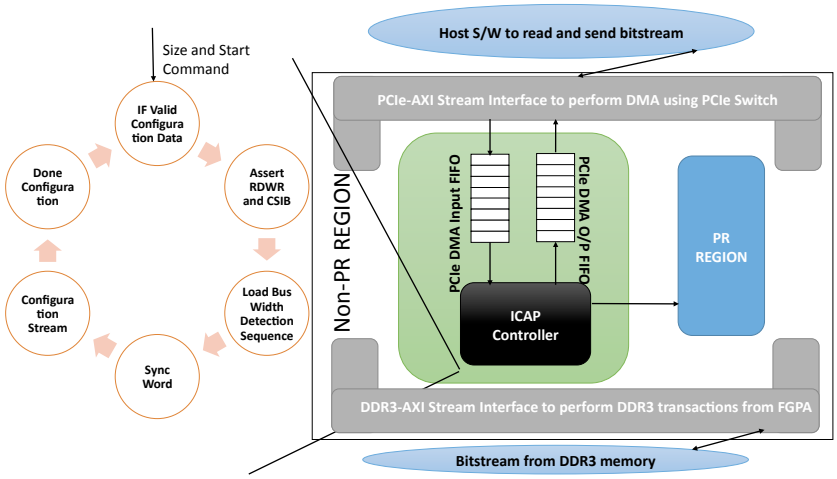

FIGURE 4.8 – ICAP controller

4.4.3 Generalizing the concepts

In the SPMD architecture, the parallel reconfiguration on 3D FPGA will be managed by the execution model. Hence, in addition to the parallel delivery of partial bitstreams to different reconfigurable regions, the execution model has a range of mechanisms that significantly improve the performance of systems in terms of size, power and execution time. As shown in figure 4.9, the model takes some information as inputs, then performs four main operations that lead to realization of the needed partial reconfigurations. The model inputs include the target FPGAs characteristics such as the number of ICAPs, the number of identical regions to reconfigure, the bitstream of the IP that we want to broadcast, and the localisation of this bitstream.

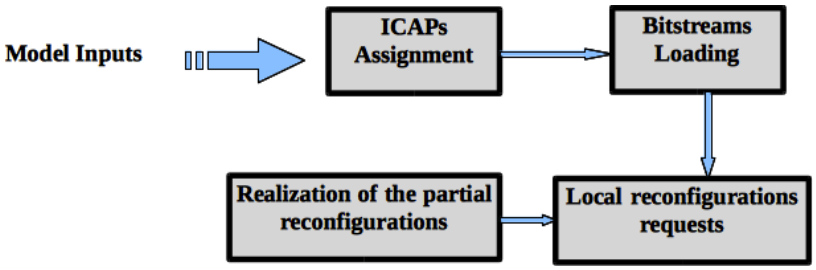

FIGURE 4.9 – Main steps for generic parallel reconfiguration model

As we target a general and flexible model that will be able to support a large range of FPGA devices, several cases may appear at the first stage i.e., ICAPs assignment as shown

in figure 4.9. In fact, the target device may contain N number of ICAP configuration ports less than the number of regions to be configured, C. This is the simplest possible situation because all the partial reconfigurations can be realized in a parallel way for sure. Thus, we assign one ICAPs for each reconfiguration. In this case the remaining non assigned ICAPs will still be in the idle state. In the second case, the FPGA does not contain enough ICAPs to reconfigure all the regions in a parallel manner. Thus, in our model, we assign X regions to the $N-1$ first ICAPs and $C-(X(N-1))$ regions to the last ICAP (X or X + 1 depending on the values of C and N) with $X = CdivN$ (div is the integer division). Such an assignment ensures a balanced repartition of reconfiguration workload on the available ICAPs. In this situation, the reconfigurations assigned to a given ICAP are realized sequentially. Our proposed prototyping environment allows to emulate these different cases.

After the ICAPs assignment step, we start loading the bistreams from the local DDR3 or the shared memories and dispatching each bistream to its assigned ICAP by storing it in a local memory accessible by the ICAP. As an example, we have used the PCIe FIFO in our system to transfer the bitstreams as shown in figure 4.11. We recall that the capability of our system to communicate one-to-many and one-to-all enables to share bitstreams between different layers. Two possibilities may occur in this step. First, bitstreams corresponding to the implementation of each IP on any reconfigurable region are available in the external memory : in this case appropriate bitstreams are simply selected and loaded in the local memories. Second, only one bistream version is available for each IP : in this case a relocation step is performed before dispatching bitstreams to the local memories. This relocation requires the modification of a given bistream that was planned for a given region in order to adapt it to another region. Such a modification is quite simple and easy to perform as the changes are still minor and involve only few bytes (translation operation).

Once all bistreams are ready and inside the local memories, reconfiguration requests are sent to the ICAPs. We recall that using the bus-based communication mode, we synchronize the start of a parallel reconfiguration in our system. Each ICAP receives the commands to perform all reconfigurations assigned to it, one by one from a local arbiter. A reconfiguration request can not reach an ICAP before it finishes the previous reconfiguration assigned to it. After receiving a reconfiguration request, each ICAP writes a partial bitstream into the target reconfigurable region.

4.5 Experimental results

As stated before, the experimental setup consists of a general purpose processor with a PCIe x16 slot where we insert our system from Pico Computing [42] as shown in figure 4.10. Our objective is to build a scalable secure video encoding encryption application respecting a massively parallel dynamically reconfigurable execution model and to emulate the next generation 3D FPGA by using several 2D FPGAs. Our application will be designed using mainly 3 coarse-grained IPs : the H.264 encoder, the AES (Advanced Encryption Standard), and the RTEA (Ruptor's Tiny Encryption Algorithm).

4.5.1 Application scenario

We illustrate an industrial high-speed FMC based data acquisition serial wide band recorder. The recorder itself is a serial Front Panel Data Port (sFPDP) IP core with 4 capture

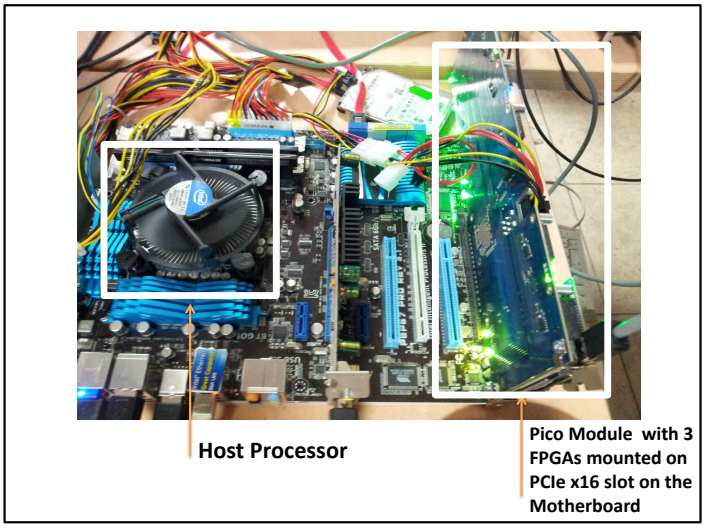

FIGURE 4.10 – Modular Multi-FPGA board mounted on the host processor system via PCIe

channels as shown in figure 4.11 and each channel operates at a 2.5Gb/s line rate. This means that at the application level the payload throughput is approximately 1 GB/s for 4 channels. The application is a secure H.264 video encoding and encryption application. A H.264 video encoder encodes the captured RAW video data, and we use for example, the Advanced Encryption Standard (AES) encryption algorithm to encrypt the encoded stream and store it in the hard disk. Here the encryption algorithm is assigned to the dynamic reconfigurable region and can be swapped for any other encryption algorithm during runtime. For instance, we can use RTEA which is a lighter and more faster encryption algorithm than AES, which usually serves as an industrial standard encryption.

In this setup, there are several tasks that are executed in parallel. First, each instance of H.264 encoder running on a different FPGA, captures RAW video data from a different source and produces an encoded video stream in parallel. Second, the process of writing back the encrypted stream to the host happens in parallel on all the FPGAs. Finally, all the three encryption algorithms can be swapped at runtime on all the FPGAs if needed by the application. And all these tasks are carried out using PCIe stream-based communication and PThreads as described in the previous section. In this way, we have emulated a SPMD based execution model on a 3D FPGA by using several 2D FPGAs.

4.5.2 FPGA resource utilization

Table 4.1 shows the resource requirements for different design scenarios in a single FPGA. The percentage of resource utilization has been mentioned in all the scenarios. While trying to implement 3 encoding channels and 3 encryption channels on a single FPGA, the design failed to meet the resource requirements for BRAMs. Therefore we simply ignore this scenario as it is not feasible in a single FPGA. For the others, depending on the design scenario, the device utilization changes. The only thing that has to be noted is that, for designs with more than 1 channel, the wrapper module should also be duplicated according to the number of channels. The wrapper consists of the logic for the operation of rest of the system, except the encoding/encryption core. For example, for a 3 channel H.264, we have to

	Slice Registers	Slice LUTs	IOBs	BRAMs	DSP	ICAP
H.264 Encoder core	2,779 (0%)	5,164 (2%)	90 (18%)	2 (0%)	2	0
Encryption Core	14,767 (3%)	10,603 (5%)	389 (77%)	0 (0%)	0	0
H.264 Encoder + Wrapper 1-channel	52,686 (12%)	38,627 (18%)	202(40%)	120 (26%)	2	0
H.264 Encoder + Wrapper 3-channels	125,497 (31%)	94,050 (47%)	202(40%)	287 (62%)	6	0
Encryption + Wrapper 1-channel	31,831 (7%)	21,061 (10%)	81(16%)	67 (15%)	0	0
Encryption + Wrapper 3-channels	84,358 (21%)	56,359 (28%)	81(16%)	157(36%)	0	0
H.264 Encoder + Encryption + ICAP + Wrapper	89,696 (22%)	61,525 (31%)	225(45%)	135(30%)	2	1(50%)
H.264 Encoder + Encryption + ICAP + sFPDP + Wrapper	112,104 (28%)	82,703 (41%)	280(47%)	138(31%)	3	1(50%)
H.264 Encoder + Encryption + Wrapper 1-channel each	67,301 (16%)	49,491 (24%)	202(40%)	120 (26%)	2	0
H.264 Encoder + Encryption + Wrapper 3-channels each	N/A	N/A	N/A	N/A	N/A	N/A

TABLE 4.1 – Area utilization for different FPGA designs scenarios in 1 FPGA

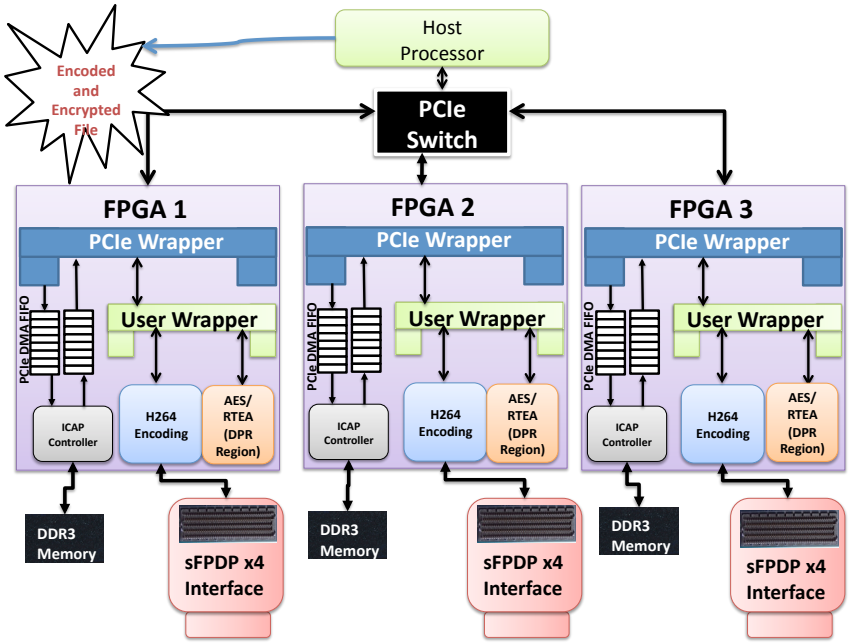

FIGURE 4.11 – Video encryption based on SPMD execution model

duplicate the AXI interconnect slaves 3 times to access the DDR3 memory. The arbitration between the controllers will be taken care of by the AXI interconnect switch. For the setup shown in figure 4.11, we have integrated a H.264 encoder, encryption core, an ICAP controller and a high-speed data acquisition sFPDP x4 IP core in a single FPGA. It consumes about 50% of the resources on a single FPGA. This is an example for a SPMD execution model on 3D FPGA because, as mentioned earlier, each FPGA represents a layer of the 3D FPGA and we have distributed each instance of the SPMD model on different layers. This setup serves different benefits such as, distributing a redundant architecture over multiple FPGAs gives the flexibility of modularity and serves to be more power efficient.

4.5.3 Application profile on the hardware

For the above application scenario, we have calculated the execution times for encoding and encryption on the hardware along with the time taken for inter-FPGA communication. First, we have measured the execution times for encoding and encrypting 90 frames of RAW video on the hardware separately. Then we measured the overall execution time for up to 90 frames including encoding, transfer of encoded data to another FPGA for encryption. From the times measured, we have extracted the time taken to transfer the encoded data from 1 FPGA to another, in order to perform encryption. Finally, we have also measured the time taken to encode and encrypt 90 frames of video on the same FPGA, instead of performing encryption on another FPGA. Figure 4.12 shows the time taken for each of the category as separate bars in the graph. From the graph, although it is obvious that the execution time increases for more number of frames, the increase is not steeply linear. This is due to the fact that we first buffer the incoming RAW frames into the DDR3 memory and then access them during encoding. The DDR3 memory controller has been designed in such way to queue

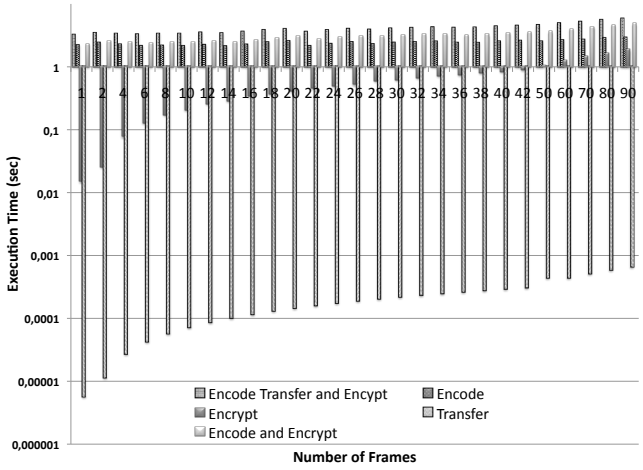

FIGURE 4.12 – Execution times on the hardware

several read addresses as burst transactions. Hence after the initial DDR access delay, we have access to 32 bytes of data every DDR cycle (250 MHz). Thus when data size increases, the execution time overshadows the initial access latency, and minimizes the steep linear increase in the overall execution time.

Furthermore, from the graph we see that it takes about less than 1 ms to transfer 90 encoded frames (about 2 MB of encoded data) via PCIe. There are three things to be noted. First, similar to the DDR3 access, after the initial PCIe transfer latency between two nodes, the PCIe stream links are capable of delivering the data at the same rate at which we produce data. Thus, as the data size increases, the transfer time is hidden by the execution time. Therefore we see that the ratio of overall execution time between bar 1 (encryption performed on another FPGA) and bar 4 (encryption performed on the same FPGA) decreases. Second, it is also possible to buffer the encoded stream in the DDR memory, perform a partial reconfiguration to swap the encoder with an encryption algorithm, and encrypt the buffered data. Swapping avionic communication protocols during run-time have shown performance gains [40]. Although this is feasible, it is not recommended due to the fact that the tools are not certified to be used for partial reconfiguration in several critical industrial applications. Hence, due to certification and stability reasons, these applications can not fully exploit the benefits of partial reconfiguration. Furthermore, partial reconfiguration will also not be suitable for streaming applications where the data flow is continuous. Finally, having 3 encoding and 3 encryption channels on the same FPGA is also not possible because, having 3 encoding channels on Kintex 7, already utilizes up to 60% of the device resources as shown in Table 4.1. Furthermore, the availability of the FMC I/O modules also dictates how we partition the application as stated earlier. Thus there is a strong need to wisely distribute the architecture in a high-performance application thus avoiding extensive redesign times.

4.5.4 Host to FPGA bandwidth

The graph in figure 4.13 shows the time taken to transfer different data sizes from host to FPGA and vice versa via x16 PCIe streams. There is a constant linear increase in the time taken with respect to data size. However, for very small data sets, the time taken to read from

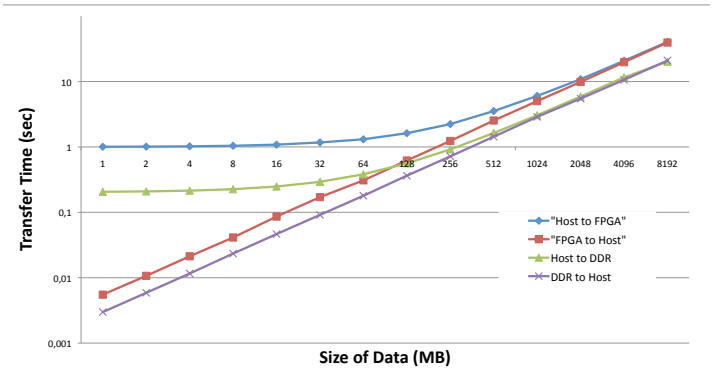

FIGURE 4.13 – Data size vs transfer time

the FPGA is greater than the time taken to write to the FPGA. This is due to the initial delay in the loop-back to happen. However, this initial delay is hidden when the transfer time is much greater. In theory the maximum bandwidth of x16 PCIe per direction is 8 GB/s i.e., 16 GB/s in total (Tx+Rx). The time was calculated using software timer including the overhead of host software function calls, and the overhead of API loops to write several GB of data. This is due to the limitation of the software API, that only allows writing 8 kB of data per call. Using the times measured in figure 4.13 the bi-directional bandwidth is calculated to be approximately 1.1 GB/s in our system with x16 PCIe lane width.

4.5.5 Host to memory bandwidth

The host to memory transfer time for different data sizes was also measured from software and shows similar trend as that of host to FPGA transfer time as shown in figure 4.13. The time increases linearly with respect to the data size. Except for smaller data sizes, where the initialization time overshadows the transfer time. Hence, for smaller data sets, the writes to DDR3 which always happens first, take more time than reads from DDR3. From figure 4.13, we have calculated the uni-directional bandwidth between host to DDR or vice versa to be 0.4 GB/s. Since we can write and read from different DDR3 locations at the same time using PThreads (without coherency problems), the bi-directional bandwidth between host and DDR memory is calculated to be about 0.8 GB/s.

4.5.6 FPGA to FPGA bandwidth

One of the main features of the system is the high-speed point-to-point full duplex links between two nodes. We have measured the size of data that can be transferred between two FPGAs at a given point of time. We have measured this bandwidth to know the time taken in cases where there is a need to transfer bitstream from one FPGA to another FPGA without host intervention. The data is plotted in figure 4.14. The transfer time was measured from the software. When the PCIe link is ready to accept data, it asserts the ready signal of the PCIe stream, and then the firmware can send 16 bytes of data in one clock cycle (250MHz). The number of 16 bytes of data transferred was calculated from the software by keeping track of the hardware counter at two discrete time points. The average net unidirectional bandwidth between two FPGAs is calculated to be about 3.2 GB/s, which makes the overall net

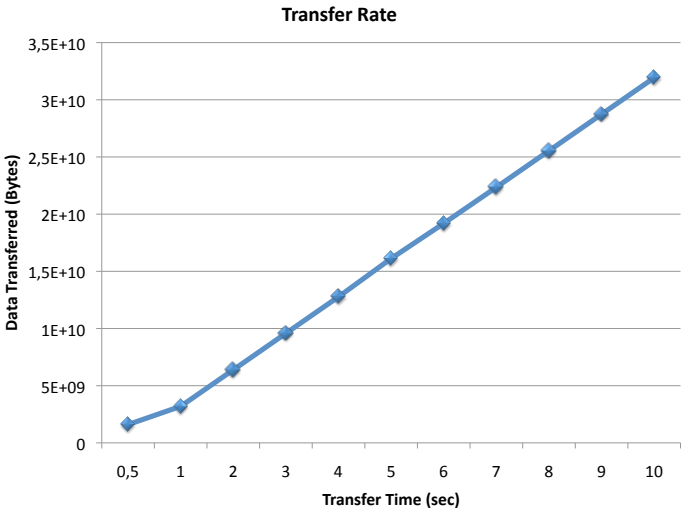

FIGURE 4.14 – FPGA to FPGA transfer time

bidirectional bandwidth to be 6.4 GB/s. Thus we are able to attain very high reconfiguration throughput in the next generation 3D FPGAs using our current emulation model.

4.5.7 Reconfiguration time

We have measured the time taken to reconfigure a single FPGA. As seen in figure 4.15, as the size of the bitstream increases, the reconfiguration time also increases. We also notice that the reconfiguration time is almost the same for both the cases of reconfiguring from the stream, and reconfiguring from the DDR3. This is because, we measure time in the software once we broadcast the start reconfiguration command. By this time, the bitstreams are already in the stream DMA FIFO ready to be read without interruption. Therefore, the overhead of reading the bitstream from the hard disk is not seen in the reconfiguration time in the graph. We observe that the reconfiguration time taken on one FPGA is the same as that taken on all the FPGAs. Therefore, the reconfiguration time is constant irrespective of the number of FPGA layers reconfigured. In this way, we verify that the reconfiguration happens in parallel for all the layers of the FPGA.

4.5.8 Bandwidth and throughput analysis

In this section we analyse the bandwidth and throughput of the sFPDP data acquisition protocol and the processing throughput of the H.264 video encoder. We have used a sFPDP protocol with 4 channels on each FPGA. Each channel transmits and receives using a Xilinx GTX tranceiver running at 2.5Gbps line rate. Taking a 20% overhead on 2.5 Gbps we get a maximum bandwidth of 250 MBps. However, the sFPD protocol itself gives a throughput of 247 MBps taking into consideration frame over heads of the protocol. And we have measured a peak throughput of 245 MBps on the software timer. This gives us an overall throughput of 980 MBps for 4 channels on each FPGA. On the other hand, our H.264 encodes 1080p HD resolution images at 20 FPS. This approximately gives the encoder a peak throughput of 39 MBps. Thus leaving the acquisition bandwidth of a single channel of sFPDP to be more

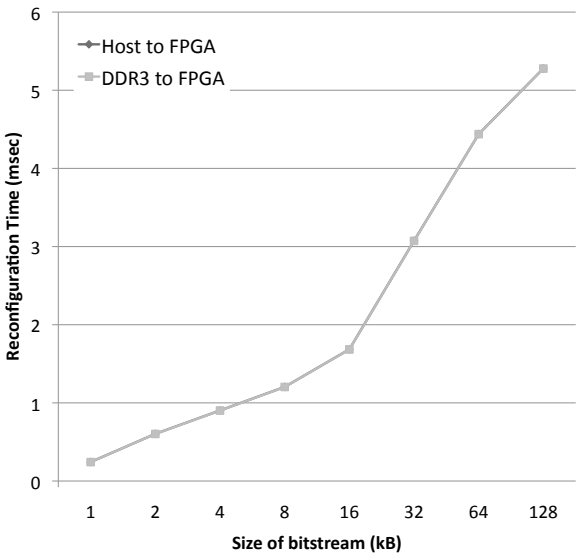

FIGURE 4.15 – Reconfiguration time vs bitstream size

than sufficient for the encoder to be working at its peak throughput. However, we have used a x4 sFPDP only as an example to illustrate the acquisition rates possible using a FMC based high-speed data acquisition protocol.

4.6 The HoMade processor

Our contributions to the field of dynamic reconfiguration and massive parallelism are centred around a softcore processor called ***HoMade***. It is an ultraRISC stack-based processor, with only 12 instructions, used essentially for controlling the execution flow (i.e. jumps, procedure call/return, master/slaves invocation...). One other instruction is used to trigger IPs via their identifier (ID). The IPs can be arbitrary complex, including ALU functions, register files, load/store units, or even other processors. A user pre-selects a set of IPs that will at some point be used by the application, to be invoked at runtime by the IP trigger instruction. Thanks to the partial dynamic reconfiguration offered by recent FPGAs, the same IP ID can be used with software (HoMade executable code) and hardware (VHDL) IP instantiations. HoMade also has an instruction that writes in its program memory, thereby dynamically modifying the program being executed. HoMade is thus a reflective processor that extends intercession to hardware [69]. There are several benefits of the reflection are :

- a reduced silicon surface, because only the IPs in use are instantiated on the FPGA ;
- a reduced power consumption for the same reason mentioned above ;
- a reduced execution/response time, by switching from software code to hardware IPs at runtime.

These savings are important for embedded systems that often need to operate with limited resources.

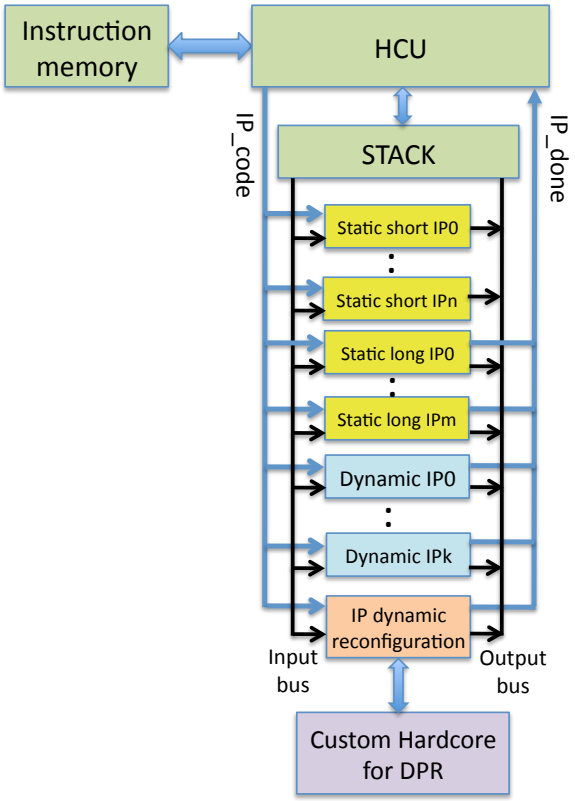

FIGURE 4.16 – General view of the HoMade architecture : our processor is mainly composed of the instruction memory, the control unit, the stack memory, and a number of IPs

4.6.1 The architecture of the HoMade processor

HoMade is an ultraRISC stack-based processor that can be considered as an IP integrator. Mainly, it is composed of an HoMade core (instruction memory, stack, and a control unit) and a set of variable IPs for running a given application. ***In HoMade, everything is an IP***. Thus, an HoMade implementation can instantiate and integrate the appropriate IPs adapted to that application. Pushing to the extreme for the principle of customization, HoMade offers neither calculation unit nor memory structures (for data) ; therefore the instruction set will be more reduced and almost dedicated to the instructions for the control flow. Before realisation on FPGA, it will be necessary to add your IPs otherwise there is nothing to do with your processor. In other words, by default, HoMade does nothing alone because there is neither ALU nor register, nor memory data, etc.

How to program a processor without ALU, registers or load/store instructions ? In fact, one and only one instruction can trigger a particular IP from 2047 possible instantiations. If a particular application requires some registers, an IP will be offered for that. If an application requires a memory, an IP will be offered for that based on the FPGA or the board certainly. If an application requires an ALU, an IP will be implemented for that and so on. Hence, before you write and run a program, you should add your own IPs or use those proposed in the default HoMade library [33].

33. https ://sites.google.com/site/homadeguideen/ip-library

As shown in figure 4.16, all the IPs can read or write from/to the stack : no address register for their in or output ports. The stack has three registers on top to allow reading or writing in the same cycle. An IP can perform at most 3 pop or 3 push in the same cycle. The number of pop or push operations is determined by the HoMade Control Unit (HCU in figure 4.16) regarding four bits in the code of the IP instruction.

The HoMade instruction set : The processor has 12 instructions in 16-bit format.

- 4 jump instructions : absolute branch, relative branch and 2 conditional relative branch. The condition is evaluated using the value at the top of the stack compared to the zero value. With these four instructions, the compilers are able to generate the code for all the control structures, at least for those of the HiHope language developed in our team as an extension for the Forth language.
- 2 function call instructions : CALL and RETURN use absolute branch in the memory instruction. The HoMade HCU has a specific stack for nested function calls.
- End of execution instruction : the HLT instruction stops the HoMade processor in its current status and reports the end of activity using one of the output pins of the processor.
- 2 instructions for parallelism : the HoMade master executes two consecutive instructions to perform parallel computing on a set of slaves. The SPMD instruction triggers the executable code stored in the instruction memories of the salves (the same starting address of the program is the only constraint). While the WAIT instruction waits until all slaves validate the termination of their execution.
- IP instruction : it can trigger an IP from 2047 possible IP. It also specifies the actions to be taken on the stack (Push / Pop) during the execution of this IP. The IPs are divided into two groups : short and long according to the execution time is equal to or greater than one clock cycle. As shown in figure 4.16, long IPs use the IP_done signal to notify the end of execution.
- Reflection instruction : the WIM (Write In Memory) instruction is the only instruction that can write in the instruction memory with strict constraints on the write addresses. It allows the change of the executed program while it is running. It will be useful for the dynamic reconfiguration "hardware/software" by allowing for example to replace a function call by a particular IP instruction.
- LIT instruction : the twelfth instruction is the LIT instruction which can manipulate 12-bit immediate values by storing them on the top of the stack.

4.6.2 Heterogeneity in HoMade

As stated before, using the heterogeneity in embedded systems was emphasised by the higher requirements in terms of performance and energy efficiency. Almost of existing solutions tackle this problem by integrating different computing nodes which leads to more complexity in the development phase. In the HoMade processor, we offer the possibility to integrate IPs with different granularities starting with elementary operations (adder, data memory access, I/O, etc.) and stretches out to a coarse-grained hardware accelerator (Fast Fourier transform) or even a softcore processor (handled as a slave IP regarding an HoMade master). Between a full software implementation (several IPs are needed) and a custom hardware (one IP), there are a multitude of solutions. Hence while developing an application, the designer can choose the appropriate granularity level for each task depending on the required performance, the power budget, the Quality of Service (QoS), etc. In this way, the

heterogeneity is introduced in HoMade in addition to the parallelism and dynamicity as it will be detailed in the next subsections. We note that, whatever the granularity of the IPs, they are integrated and scheduled in the same way in the HoMade processor. The Listing 4.1 gives an example of a software implementation of the Fibonacci algorithm relying on several IPs essentially the IP_Stack necessary to manipulate the data at the top of the HoMade stack (rotate, duplicate, etc.) and the IP_ALU for arithmetic and logic operations. These two IPs are of type short and available in the default HoMade library. This example allows to read a number N from the input switches, to run the Fibonacci sequence with N terms, and to show the result and the number of execution cycles on the 7 segment display. We note that the IP_Stack is a set of IPs to perform elementary operations on the HoMade stack. In addition, our program explicitly calls the IP_Datastack that functions as a stack for nested control structures like DO LOOP and FOR NEXT (Listing 4.1, lines 17 and 23).

```
1  program
2  : read // A function to read the number of terms (N) in the Fibonacci
3         // sequence from the input switches
4          $ 1f // A mask value to check which push button is pressed
5                  // here any push button is accepted
6          btnpush // IP to start the program
7          switch  // Call for the switch IP to read the input value
8          LedDup  // Call for the led IP to show the input value
9  ;
10 : fibo_soft // A function to calculate the Fibonacci sequence
11         $ 0     // Put the constant 0 on the top of the stack
12         $ 1     // Put the constant 1 on the top of the stack
13         rot         // Call for the stack IP to rotate the top
14                 //  three elements of the HoMade stack.
15         $ 3     // Put the constant 3 on the top of the stack
16         -  // Call for the ALU IP to decrement 3 from N
17         for //  N is pushed to the data stack, and each iteration,
18             //  if N != zero then N is decremented by 1, else exit.
19                 dup // Call for the stack IP to duplicate the top
20                    //  element of the HoMade stack.
21                 rot
22                 +  // Call for the ALU IP to get the next Fibonacci term
23         next   // Check for the branch condition from the data stack
24         swap
25         drop
26 ;
27
28 : main  // The main program
29         begin
30                 read // Call for the read function
31                 ticraz // Set the timer IP to zero
32                 fibo_soft // Call for the fibo_soft function
33                 tic // Read the number of cycles from the timer IP
34                 7seg // Show the execution time in cycles on the display
35                 $ 1f
36                 btn
37                 7seg // Show the last Fibonacci term on the display
38         again
39 ;
```

```
40 start
41         main
42 endprogram
```

Listing 4.1 – Software implementation of the Fibonacci algorithm

In the Listing 4.2, we give an example of a hardware implementation of the Fibonacci algorithm using one dedicated hardware IP. Compared to the software implementation, we are calling an IP named fibo (Listing 4.2, line 13) to carry out the Fibonacci sequence. This IP consumes the value N from the top of the stack and store the result again on the stack after the end of execution.

```
1  program
2  : read
3          $ 1f
4          btnpush
5          switch
6          LedDup
7  ;
8
9  : main
10         begin
11                 read
12                 ticraz
13                 fibo // Call for the fibo IP
14                 tic
15                 7seg
16                 $ 1f
17                 btn
18                 7seg
19         again
20 ;
21 start
22         main
23 endprogram
```

Listing 4.2 – Hardware implementation of the Fibonacci algorithm

Listing 4.3 illustrates the implementation and the plug-in of the IP fibo in the processor HoMade. We note that the exposed lines of code belong to different VHDL files. Lines 1, 2, and 3 are a part of IPcode.vhd which is a package file where we define the IPcode used for different IPs in our system. For example, we define IPcode = 0b10000000011 for the fibo IP while to add it to the master/slave HoMade then the GenM_fibo/GenS_fibo is set to 1. Listing 4.3 (from Line 6 to Line 18) shows the entity description for the fibo IP such that it needs only to read one input value (i.e. Fibonacci order) and write back one value (i.e. Fibonacci number) therefore ; only the top element of the HoMade stack is connected to the IP as described in the top level module for the HoMade master shown in the listing (from Line 21 to Line 32). The fibo IP is a long IP so we are noted for the end of execution when the IPdone signal is set high.

Inside the IP fibo, we can observe that the IP waits in the idle state till the input IPcode matches with Mycode as shown in Listing 4.3 (Line 38) so that it can start calculating the Fibonacci number as illustrated from Line 46 to Line 57.

```
1  constant GenM_fibo : std_logic :='1';
2  constant GenS_fibo : std_logic :='0';
3  constant IPfibo    : code := "10000000011";
4
5  // Entity description for the fibo IP
6  entity IP_fibo is
7  // mycode is  the IPcode chosen for the fibo IP
8          generic ( mycode : std_logic_vector(10 downto 0) );
9          port ( clk : in std_logic;
10                    reset: in std_logic;
11 // read the top level element of the stack
12                 Tin : in  std_logic_vector (31 downto 0);
13                 IPcode : in  std_logic_vector (10 downto 0);
14 // write on the top level element of the stack
15                 Tout : out  std_logic_vector (31 downto 0);
16 // IPdone is set to 1 when execution is completed
17                 IPdone : out  std_logic);
18 end IP_fibo;
19
20 // The IP fibo connected to the HoMade stack
21 fibo : if genM_fibo = '1' generate
22 Inst_IPfibo: IP_fibo
23         generic map ( Mycode =>IPfibo )
24         port map          (
25                           clk      => clock,
26                           reset    => reset,
27                           Tin      => Tin_stack,
28                           IPcode   => IPcode_stack,
29                           Tout     => Tout_stack,
30                           IPdone   => IPdone_stack
31                           );
32         end generate Mfibo;
33
34 // Verification of the IPcode
35 case current_state is
36         -- waiting for correct IPcode matching
37      when idle =>
38                 if IPcode =  mycode then
39                           next_state <= starting;
40                           rst  <='1';
41                 else
42                           next_state <= idle;
43                 end if;
44
45 // Calculating the Fibonacci number
46 process  (clk, init)
47         begin
48                 if clk'event and clk='1' then
49                           if init ='1' then
```

```
50                                      fib <= x"00000000";
51                                      fib2<= x"00000001";
52                              else
53                                      fib2 <= fib2 + fib;
54                                      fib <= fib2;
55                              end if;
56                      end if;
57  end process;
```

Listing 4.3 – Hardware description of the fibo IP

We have implemented both the software (fibo_soft) and the hardware (fibo_hard) using the HoMade processor on the Nexys 3 board (Spartan6-XC6SLX16) in order to compare the performances. By analysing the execution for different Fibonacci numbers, we can note that normally for one iteration fibo_soft needs 9 clock cycles while fibo_hard requires only one clock cycle. This is mainly due to the fact that fibo_soft is composed of a set of elementary IPs where each IP needs one cycle in addition to the control instructions. However, fibo_hard makes profit from the execution of concurrent operations to improve the speed. Table 4.2 shows the gain by using fibo_hard according to the Fibonacci order. The achieved gain can be more important with more complex functionalities. As a conclusion, the software approach allows to develop rapidly the program based on IPs with the cost of less performance efficiency. While the hardware approach leads to better performance with more development effort.

Fabonacci order	Number	Fibo_soft (in cycles)	Fibo_Hard (in cycles)
4	2	34	7
12	89	106	15
20	4181	178	23
23	17711	205	26
24	28657	214	27
25	46368	223	28

TABLE 4.2 – Execution time in cycles for different Fibonacci order

4.6.3 Reflection in HoMade

HoMade is designed to support dynamic configuration : Software/Software, Software/Hardware, Hardware/Hardware. To implement these features in the processor HoMade there is a WIM (Write in Instruction Memory) instruction that proposes to change the content of the program at runtime. This change is non-preemptive operation controlled by the processor HoMade. Reflection as a hardware concept for HoMade processor : this new instruction is dedicated to changing the program memory at runtime. Of course, it is necessary to remember that the HoMade processor is following the Harvard architecture with a separated instruction and data memory (this data memory exists only if a memory IP has been instantiated during the processor configuration). The WIM instruction allows writing in the program memory (for master or slave) a sequence of three instructions aligned on a 64-bit word. The WIM instruction must be aligned at the beginning of a 64 bits-word in the me-

mory. Only the interval between addresses 1&00 to 111111111111&00 are accessible by this reflection instruction. The WIM instruction contains explicitly the address (the 12 most significant bits of the address). The first 48 bits of the targeted 64-bit word represent the 3 instructions to be written in the memory while the fourth instruction is not modified by the WIM and it should be initialized either by RETURN or HLT according to CALL or SPMD instruction.

The instruction WIM will be used mainly to support the concept of Virtual Component (VC). In fact, VC is a high level concept defined in order to encompass several functions which correspond to the different implementations (hardware and software). At runtime, a decision can be taken to modify the program in the instruction memory and to choose a given implementation according to the environmental parameters, the power budget, etc. The Listing 4.4 gives an example of implementing the reflection mechanism for the Fibonacci algorithm in order to swap dynamically between the software and the hardware implementations introduced previously. In this program, we select a given implementation according to the switch value chosen by the user. In line 8 of Listing 4.4, we define a VC named fibo_dyn interpreted as a label of the memory address where the instructions of the selected implementation (fibo_soft or fibo_hard) will be written. In our program, the key word dynamic is interpreted as a WIM instruction. Hence, the line 38 corresponds to the following instructions WIM CALL NNNN nnnn that will build a dynamic indirection to the fibo_soft call by storing CALL NNNN nnnn RETURN in the instruction memory. We note that NNNN nnnn is the address of the fibo_soft function. In the same way, the line 42 of Listing 4.4 corresponds to the following instructions WIM fibo RETURN NULL that will build a dynamic indirection to the fibo_hard (the hardware implementation) call by storing fibo RETURN NULL RETURN. In this way, reflection is introduced in HoMade to support dynamic reconfiguration "software/hardware" by allowing for example to replace a function call by a realised IP or vice-versa.

```
1
2  :fibo_hard // A hardware implementation of the Fibonacci sequence
3          fibo return null
4  ;
5
6
7  program
8  VC fibo_dyn // A Virtual Component definition
9  : read
10         $ 1f
11         btnpush
12         switch
13         LedDup
14 ;
15 : fibo_soft // A  software implementation of the Fibonacci sequence
16                 $ 0
17                 $ 1
18                 rot
19                 $ 3
20                 -
21 for
22         dup rot +
23 next
```

```
24 swap
25 drop
26 ;
27
28 : main
29
30 begin
31         read
32         swap
33         $ 1
34         =
35         if  // test of the pressed switch
36                 // WIM instruction for a dynamic indirection to the
37                 //software implementation fibo_soft
38                 ['] fibo_soft fibo_dyn  dynamic
39         else
40                 // WIM instruction for a dynamic indirection to the
41                 //hardware implementation fibo3IP
42                 ['] fibo_hard fibo_dyn  dynamic
43         endif
44         ticraz
45         fibo_dyn // Call for the selected implementation
46         tic
47         7seg
48         $ 1f
49         btn
50         7seg
51 again
52 ;
53 start
54         main
55 endprogram
```

Listing 4.4 – A dynamic Hardware/Software implementation of the Fibonacci algorithm

4.6.4 Dynamic reconfiguration in HoMade

As stated before, the WIM instruction allows to switch dynamically between Software/-Software, Software/Hardware or Hardware/Hardware configurations. When a hardware implementation is involved we can perform a dynamic instantiation of IPs during the execution of the algorithm by using the DPR feature on a predefined reconfigurable regions as shown in figure 4.16, which can improve power and area utilisation efficiency. As the instantiated components have a generic interface, they can be plugged in the HoMade processor easily during the runtime. In HoMade, there is no a specific instruction for partial dynamic reconfiguration management ; however, the designer can create his own IP that controls the dynamic reconfiguration of IPs. In Section 4.4, we detailed a parallel reconfiguration model for these IPs speculating the emergence of 3D FPGA technology. Dynamic IP instantiation is handled by an HoMade master in order to reconfigure one or several regions. In fact through an SPMD instruction, the master can initiate a parallel reconfiguration on a set of active slaves according to the technological features (mainly the number of available ICAPs)

of the FPGA (3D IC) or the board. To perform this reconfiguration, the IP needs the address of the bitstream that can be popped from the top of the stack. In order to use efficiently the memory, our current work focuses on a runtime relocation mechanism of bitstreams while targeting regular reconfigurable regions.

4.6.5 Parallelism in HoMade

In the HoMade processor, two instructions SPMD and WAIT can trigger a set of parallel HoMade slaves with a synchronized end of execution. The active slaves will start the execution when the signal StartCPU is activated. The master sends 13-bit address to the slaves through the StartAddress port as shown in figure 4.17 then it can waits for the end of execution of the slaves (WAIT instruction).

At the active slaves level, the received address represents the start of the SPMD program. At the end, the HLT instruction will trigger the activation of the ORTREE signal. When all the slaves assert their ORTREE signal, the SPMD_done signal is activated to allow the master continuing its execution as shown in figure 4.17.

The WIM instruction can modify this code without changing the final HLT. We can for example place a CALL on 3 words followed by a HLT to obtain a SPMD instruction. We can also place one instruction followed by a HLT to emulate a SIMD mode (at least 4 cycles are required : SPMD + Instr_SIMD + WAIT + HLT). The combination of the WIM instruction (for hardware/software) and the SPMD instruction on different subset of slaves leads to the Multi SPMD execution model. The activity management mechanism selects a subset of the slaves contributing in the SPMD function. A flip-flop can be configured by an IP from the master or from a slave. These IPs are available in the basic library of the HoMade. Managing the communications between the master and slaves depends on the network topology and are realised also through dedicated IPs. The HoMade library offers a generic solution for a 2D torus grid.

4.7 Discussion

The Summary. In this chapter, we presented our contributions in the field of parallel and dynamic reconfigurable computing targeting high performance and adaptive embedded systems. First, we proposed an efficient execution model (*massively parallel dynamically reconfigurable*) that deals with the high requirements of performance, power efficiency, and adaptivity. We detailed the softcore HoMade processor that implements the above requirements respectively through the parallelism, the heterogeneity and the reflection concepts. Furthermore, we offered the appropriate Domain-Specific Language (DSL) to handle these concepts at the same level. Our processor has the benefit of ultra-reduced instruction set advocating *an all IP* paradigm, in order to favour the reuse and to reduce the silicon area, the power consumption and the execution/response time while switching between software and hardware implementations.

At the technological level, we anticipated the emergence of 3D FPGAs and we speculated the partial reconfiguration possibilities that these devices will offer for high-density applications. Indeed, as a 2D technology fact, reconfiguring such high density devices in a linear and sequential way will render the DPR feature of the FPGAs inefficient, specially for architectures featuring massively parallel dynamically reconfigurable execution model. To

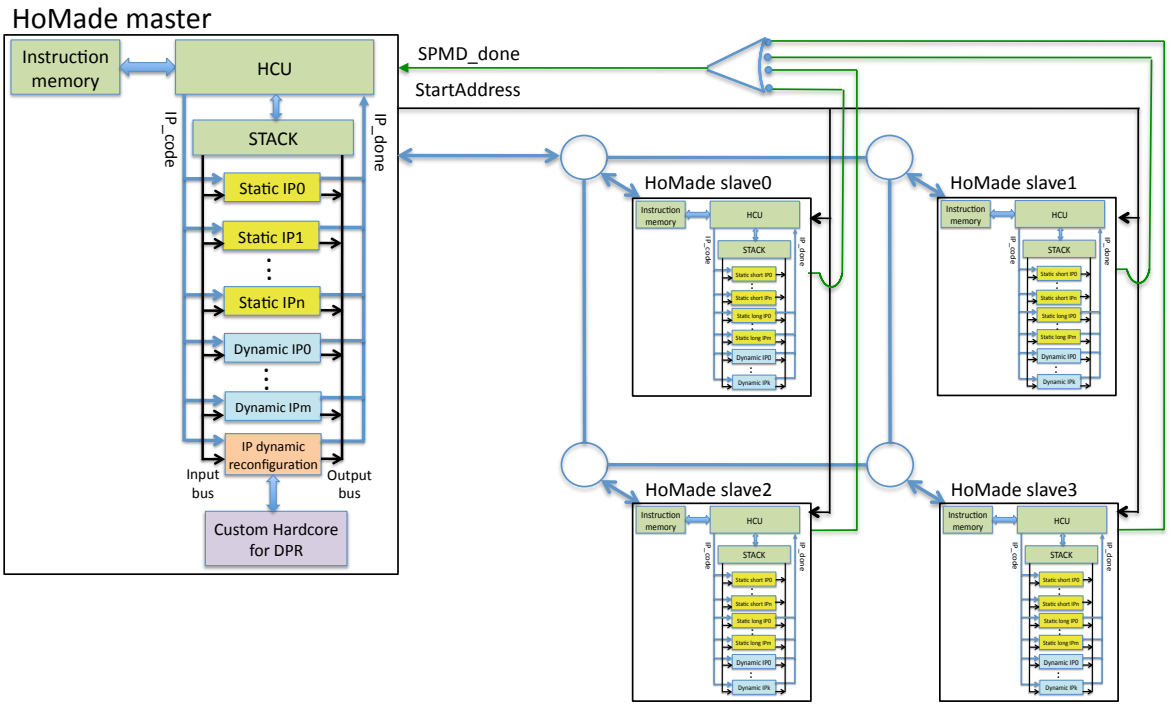

FIGURE 4.17 – Example of an HoMade-based parallel architecture

address this challenge, we proposed a partial reconfiguration model for next generation 3D FPGAs well-traced on the execution model in order to reconfigure in parallel a subset of the computing nodes. To validate our approach, we rely on a multi-FPGA based architecture that can support parallel communication capabilities with two or more FPGAs at the same time. We have then measured several performance metrics to show the efficiency of such a parallel and partial reconfiguration model.

Limitations. The proposed approach still needs more experiments with real application scenarios featuring I/O sensors. We already studied a generic data distribution model for parallel architectures to deal with high video I/O throughputs [12]. However, we need to plug this distribution system with a massively parallel dynamically reconfigurable architecture in order to evaluate the overall performances. Using a modular FPGA based computing power along with the PCIe communication channels, with several other features such as streams and PThreads, we are already able to emulate next generation 3D based reconfiguration model. However, this is just a speculation based on the existing 2.5D based FPGAs. There is still a long way to explore both in terms of technology and in terms of tools. The next generation 3D FPGAs should be able to support multiple ICAP sites on different SIC layers in order to configure several layers in parallel. The 3D FPGA ICAP should be able to operate at much higher frequency than the present generation ICAP. Finally, the tools should be able to generate partial bitstreams well suited for multiple 3D layers with proper bitstream headers. All of these improvements in the future will pave the way for convergence of massive parallelism and dynamic reconfiguration in next generation FPGA technology.

The future. A self-adaptive Multi-HoMade architecture will be the future story. Our first step is runtime performance and power monitoring. As, we will implement a wrapper around each IP in order to monitor its performance and power consumption estimation

at runtime. We will introduce dedicated sensors for capturing the local activities affecting the overall performances of one HoMade. This wrapper will be developed as an IP and integrated in the execution model of HoMade. As a part of the reconfiguration model we have to observe the multi-HoMade behaviour and to adapt the performance on chip to take care of both massively parallelism (communications and computations) and dynamic reconfiguration. To change the grain of parallelism we envisage to dynamically change the size of the grid at runtime. Adding a line or column, deleting a line or a column need to organise the regions of the FPGA in such a way the communication links are predefined and the surface is used efficiently for any IPs of any HoMade slaves. As soon as 3D FPGAs become available, Multi-HoMade could be deployed on 2D Multi-FPGA board taking benefits of all previous results.

Summary	Supervision :	- **Venkatasubramanian Viswanathan**, third year PhD student, University of Valenciennes, LAMIH, INRIA Lille. - **Karim Ali**, second year PhD student, University of Valenciennes, LAMIH, INRIA Lille. - **Wissem Chouchene**, second year PhD student, University of Lille1, LIFL, INRIA Lille.
	Journals :	
	Selected conferences :	Reconfig 2012 [40], ACM FPGA 2014 [41], Reconfig 2014 [12]
	Realization :	HoMade processor [34]
	Funding :	ANRT (CIFRE), Doctoral school grant, ERASMUS grant

Chapter 5

Conclusion et perspectives

The contributions presented in this document addressed the design of embedded systems, more generally implemented on homogeneous or heterogeneous multiprocessor or FPGA-based platforms. First, we emphasised on the usage of the reconfigurable technology in the manufacturing process of avionic systems via a seamless process considering the simulation, test, and integration phases. Second, they considered a power-aware design methodology to lead to the development of parallel and heterogeneous power-efficient systems. Third, we defined an efficient execution model for next generation massively parallel dynamically reconfigurable systems through the HoMade processor and we offered the corresponding domain-specific language. Dynamicity at the software or the hardware level was the main characteristics of these systems. For this reason, our research focused on the architectural as well as the application aspects of the system. In the different contributions, we developed the appropriate tools and prototype environments well-suited to the application domain. It is worth mentioning that these results have been achieved in collaboration with colleagues form LAMIH and INRIA Lille (DaRT and after that DreamPal). Now, we are developing a new design vision about embedded computing taking into account technological ascendancy and the requirements of nowadays sophisticated applications. Our future project will rely on the main obtained results during the last years, especially the low power design methodology and the massively parallel dynamically reconfigurable execution model implemented through the HoMade processor.

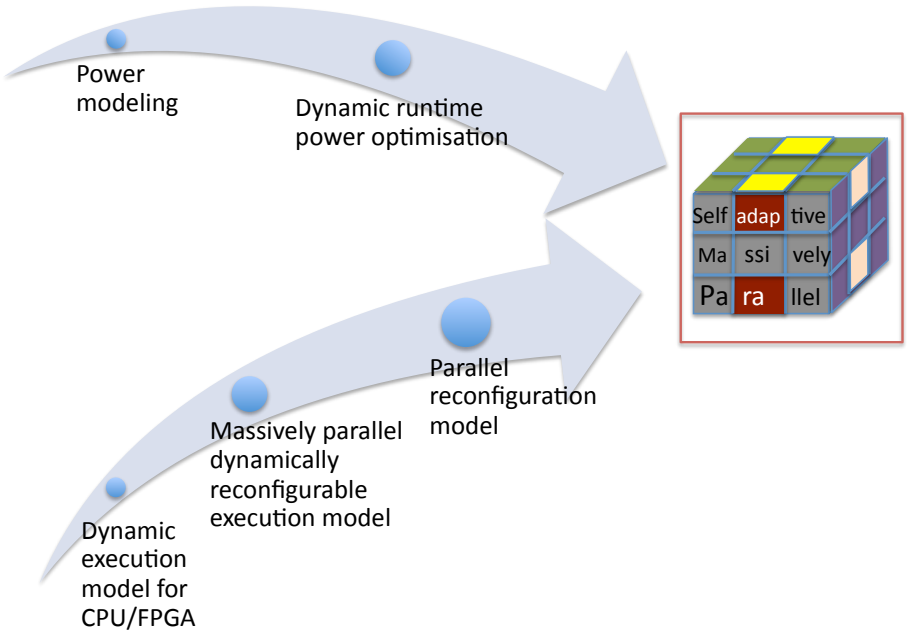

FIGURE 5.1 – Future directions towards self-adaptive massively parallel embedded systems

5.1 Overview of contributions

Based on the paradigms presented in the introductory chapter, my contributions were organized into three main steps, corresponding to the three chapters following the introduc-

tory chapter :

Chapter 2 : presented an FPGA-centric design process for avionic systems. The role of the FPGA has been redefined in the different design steps namely the simulation, the test, and the integration phases. The main contributions in this field cover the usage of heterogeneous CPU/FPGA architecture, the related dynamic execution model, and the real-time simulation environment. In this environment, we succeeded to bring intelligence through efficient heuristics that focus on the dynamic mapping of new applications at run-time, and the dynamic reconfiguration (software-software or software-hardware) to avoid the real-time constraint violation. Based on these contributions several results can be exploited while targeting our future project. First, the design methodology to obtain multitude software and hardware implementations for a given task or application [5] can be used while developing HoMade IPs. Second, the developed heuristics for runtime mapping can be adapted and used in a self-adaptive control mechanism.

Chapter 3 : presented a multilevel power-aware design methodology for homogeneous or heterogeneous Multiprocessor System-on-Chip (MPSoC). The first attractive aspect in our proposal is the global vision while considering the power consumption of the system. Indeed, we addressed the global system consumption that includes processors, memories, reconfigurable circuits, operating system services, etc. The second attractive aspect in the proposed methodology is the used power modeling approach. Actually, FLPA was used to develop power models for hardware and software components well-adapted to the system level design. these power models are integrated in the design flow in the context of multilevel design space exploration : to refine power and energy estimations, they are used in conjunction with simulation tools at different abstraction levels. The main advantage of such approach is to abstract the conventional electronic details linked to the power estimation essentially for software designers. The integration of runtime power management techniques in the design flow is another benefit of our methodology. As a cost-effective software development approach, we relied on Model Driven Engineering (MDE) to abstract the concepts and to automate the design process of MPSoC while considering the power metric. In the future, we will rely on this experience on low power design to model the consumption behaviour of an HoMade-based architecture which can be useful in the observation mechanism of a self-adaptive embedded systems. Furthermore, runtime power optimisation techniques can be coupled with the massively parallel dynamically reconfigurable execution model for better power efficiency.

Chapter 4 : exposed massively parallel dynamically reconfigurable execution model anticipating the arrival of 3D technologies. First, we defined an appropriate multi-FPGA platform that supports architectures featuring such execution model. Second, we proposed a partial reconfiguration model for next generation 3D FPGAs well-traced on the execution model in order to reconfigure in parallel a subset of the computing nodes. Third, we proposed an efficient execution model (*massively parallel dynamically reconfigurable*) that deals with the high requirements of performance, power efficiency, and adaptivity. We detailed the soft-core HoMade processor that implements the above requirements respectively through the parallelism, the heterogeneity and the reflection concepts. Furthermore, we offered the appropriate Domain-Specific Language (DSL) to handle these concepts at the same level. Our processor has the benefit of ultra-reduced instruction set advocating *an all IP* paradigm, in order to favour the reuse and to reduce the silicon area, the power consumption and the execution/response time while switching between software and hardware implementations. The results of this Chapter constitute the fundamentals of self-adaptive massively parallel

embedded systems in terms of execution and reconfiguration models that should be completed with an efficient observation and decision making mechanisms.

5.2 Perspectives : Self-adaptive massively parallel embedded systems

Today, applications in the field of intelligent transportation systems demand more than performance, power efficiency, and adaptivity. In fact, they need to be self-adaptive by the means of integrating intelligence in such way they are capable of modifying their behaviour autonomously. In September 2013, we started a new project with Airbus Helicopters that deals with an autonomous assistance system for helicopters in emergency cases. Recently, we started also a new collaboration with Induct Technology [35] in the frame of the PhD of Karim Ali in order to conceive an embedded computing system for autonomous vehicle navigation. Undoubtedly, the essential feature of systems to reconfigure themselves autonomously at run-time comes with additional complexity in the different design steps. My contributions during the last years strongly promote the convergence between low power design and dynamic execution model to reach self-adaptive massively parallel embedded systems. A self-adaptive Multi-HoMade architecture will be the future story.

Compared to the dynamic execution model presented in our work, the self-adaptive system requires a control loop based on a three-step process : 1) observation, handled by a set of monitors, 2) decision making, which analyses the observed data to adapt and optimise the system, and 3) action, which tunes the system parameters accordingly. Integrating intelligence into the circuit so that it is capable of modifying its behaviour autonomously is not a new idea [65]. But today, we have to meet all the conditions related to the technological fact (such as Dynamic Partial Reconfiguration), the execution model, and the programming language to build such circuits. In other words, we need to make the fact of self-adaptivity programmable while targeting massively parallel dynamically reconfigurable architectures. Our basic idea of *all is IP* will constitute the guiding principle of research to reach self-adaptive massively parallel embedded systems. In order to achieve this objective, our future works will follow the three main steps of self-adaptivity : observation, decision making, and action.

5.2.1 Observation

Self-adaptivity relies on the observation of different parameters such as power, performance or quality of services at runtime to analyse the behaviour of the system before decision making. In current circuits, observation is feasible with different kind of sensors like activity counters, temperature sensor, pre-build monitors [38], etc. Following the HoMade philosophy, we should implement the monitoring of theses parameters as specific IPs that can be interfaced with the monitored IPs on one side and with the decision making system on the other side. Hence, they can be activated by the HoMade programming language or instantiated at the demand of the system at runtime. We can distinguish two types of observation IPs. First, IPs build around sensors for monitoring the non-functional properties of the circuit (temperature) or the environmental parameters. Second, IPs describing a given law of power consumption or quality of service based on the capture of local activities affecting the overall performances of one IP, one HoMade, or a set of HoMade slaves.

35. http ://induct-technology.com/

Designers can add to the architecture just the needed monitors to observe a given parameter because these IPs also will consume area and power. As the monitors are considered as HoMade IPs, they can be added and removed dynamically. The main scientific challenge at this level is to determine the granularity of the monitoring IPs and how they are distributed on the architecture. In fact, a monitoring IP can be used to observe the behaviour of another IP, an HoMade processor (master or slave), or a cluster of HoMade slaves. Between a full distributed and a centralised solutions there are other possible alternatives that can offer a better balance between the complexity of the monitoring system and the granularity of observation. Our previous work in the field of distributed control in reconfigurable FPGA systems can be adapted to the context of an HoMade-based architecture [38]. In our future work, a particular attention will be given to the development of power monitors to address the runtime power efficiency issue.

Power modeling : Mapping efficiently sophisticated applications using the HoMade approach needs a knowledge about the power behaviour of the different architecture components namely the HoMade core, the IPs, the reconfiguration mechanism, the interconnection network, etc. As stated in the previous chapter, a multitude of implementations are possible between software and hardware while targeting a given functionality which can lead to different trade-offs between power and performance. Designing power efficient circuit means delivering the required performance with the minimum power budget. Hence, it becomes necessary to rely on a system level power modeling approach in order to master the consumption of an HoMade-based architecture. In Chapter 3, we already presented the Functional Level Power Analysis (FLPA) that can be used to evaluate the main functional blocks of our architecture. However, there are several scientific challenges that need to be resolved. First, we have to deal with the heterogeneity issue such as IPs of different granularities, static and dynamic reconfigurable regions, clustered architecture with different sizes (for the multi-SPMD), etc. Second, the power model should estimates the dynamic, short-circuit, and leakage power consumed by FPGAs. Third, the method for annotating the main activities that consume power should be defined. Fourth, we have to unify the power models plug-in in the HoMade architecture and how they can be interfaced with other components for decision making as it will be discussed later on. Finally, the model should be generic in that it can estimate the power for a wide variety of FPGA architectures. The developed power models will be used as a main part of the self-adaptivity mechanism in order to investigate at run time the impact of various architectural parameters.

5.2.2 Decision making

The objective of this part is to bring intelligence to the system so it can make efficient decisions, which strikes the balance between the power consumption, performance and area. The main challenge consists in selecting the appropriate implementation of each task (software, hardware, parallel, dynamic, etc.) and mapping the different tasks on the available resources (static and dynamic regions). We can distinguish the initial static mapping and the dynamic mapping at runtime. At this level, we can use the mathematical modeling and the heuristics developed in Chapter 2. However, we have to extend our previous work in order to resolve a multi-objective optimisation problem taking into account power, performance, and area. The decision making will be implemented in our architecture as IPs. As we are targeting a clustered architecture (Multi-SPMD), the reconfiguration decisions can be distributed respecting the organisation of the observation IPs. In order to guarantee that the local

decisions made by the distributed mechanism respect the global system constraints/objectives, a coordinator may be used. This distributed structure has advantages at the design and implementation levels. At the design level, handling local and global optimisation problems by separate IPs decomposes design complexity and facilitates the reuse of the observation, the decision making and the coordinator IPs. At the implementation level, the advantages of the distributed over the centralized solution are : 1) a higher parallelism of the control mechanism (monitoring, decision-making and action) and thus a higher control performance, 2) shorter connections used to collect monitoring data and thus less power consumption related to connections, 3) easier adaptation to large systems such as multi-FPGA systems, where the communication cost of centralized control solution may have a significant effect on the control performance. The main challenge at the decision making level is to characterise the available hardware resources in order to satisfy the application requirements. For this reason we will call for an abstract reconfigurable hardware model as a leverage of low level technical details.

Abstract reconfigurable hardware model : Reconfigurable computing has the benefits to offer a computing performance and power efficiency compared to processor based systems. However, based on our experience on FPGA development, we can confirm the low design productivity especially while handling dynamic reconfiguration for parallel reconfigurable architectures. Indeed, partitioning the FPGA into static and dynamic regions, storing the bitstreams, conceiving the reconfiguration controller, etc. are tedious tasks. It is necessary to abstract implementation details in order to exploit the key advantages of reconfigurable hardware and to facilitate the design of self-adaptive massively parallel embedded systems. In order to solve this challenge, we propose to develop an abstract reconfigurable hardware model that can hide low level details such as spatial placement, hardware structure and device capacity. This abstract model should ensure the matching between the system requirements in terms of logic, reconfiguration, monitoring, etc. on one side and the available resources in a given FPGA on the other side.

As a first step, we need to organise the FPGAs in the form of regular static and dynamic regions with pre-defined communication links useful to connect a reconfigurable region to an HoMade stack, to plug a new HoMade to the network of slaves, or to add a new line or column to the grid network. Since the hardware synthesis step takes significant time, it is very important to know in advance before the static or the dynamic reconfiguration process that the solution is first of all feasible in terms of hardware resources. Hence, the organisation of the FPGA chip is a very important task and can help to reduce the development time. Furthermore, this can help to characterise accurately the main architectural blocks in terms of power consumption and performance according to the area of the region, the frequency, etc. The different regions can be identified using the physical address on the FPGA. It is necessary to use a matching mechanism between the reconfiguration bitstreams and the reconfigurable regions respecting the size of each region. To do so, we can rely on indirection tables as well as bitstream relocation mechanism. In the same way, a matching mechanism should be defined between the HoMade IPs identified with ID and the bitstreams. Furthermore, the knowledge of the available ICAPs are very useful for the parallel reconfiguration model as described in Chapter 4.

5.2.3 Action :

At the action level, we already showed that it is possible to tune the voltage/frequency pairs, to migrate the code of a given task from software to hardware or vice-versa, to reconfigure at runtime a dynamic region or even several regions in parallel, etc. Given the increasing complexity of embedded systems, our approach is to consider that these actions are done through IPs that are activated by the execution model. Several action scenarios are possible : it is possible to change the grain of parallelism, we plan to dynamically change the size of the grid at runtime. Adding a line or column, deleting a line or a column needs to organise the regions of the FPGA in such a way the communication links are predefined and the surface is used efficiently for any IPs of any HoMade slaves.

Optimising embedded systems for low power consumption pushes developers to find a balance between performance and power usage at runtime. However, achieving this balance requires power models and advanced hardware power management techniques coupled with the execution model. Conventional software and hardware power optimisation techniques such as Dynamic Voltage and Frequency Scaling (DVFS) or clock gating can be used in an HoMade-based architecture. For instance, the clock of non-used dynamic regions can be disabled for reducing dynamic power dissipation. However, this needs a deep knowledge of the structure of the FPGA and its different clock domains. In line with the HoMade philosophy, the optimisation techniques can be implemented as IPs and considered as a part of the execution model. Hence, they can be handled by the programming language of HoMade at the same level with parallelism and dynamicity. We can imagine that the self-adaptivity mechanism can take the decision of activating a power optimisation IP regarding the available power budget on the battery or increasing the frequency with the help of a DVFS IP in order to achieve a certain performance regardless the power metric. As soon as 3D FPGAs become available, we will continue the deployment of HoMade-based self-adaptive embedded systems on 2D Multi-FPGA board following the above future directions.

Appendices

Annexe A

Curriculum Vitae

A.1 Identification

Nom de famille : **Ben Atitallah** Prénom : **Rabie**

Date de naissance : **13/10/1978** Grade : **Maître de Conférences** 5^{eme} **échelon**

Établissement d'affectation : **Université de Valenciennes et du Hainaut-Cambrésis (UVHC), Laboratoire d'Automatique, de Mécanique et d'Informatique industrielles et Humaines (LAMIH) UMR UVHC/CNRS 8201**

Formation :

- **2004-2008** • Doctorat en Informatique à l'Université des Sciences et Technologies de Lille (USTL) au sein de l'EPI (Equipe Projet INRIA) DaRT, mention très honorable.
- **2002-2003** • Master de recherche en Electronique, option Micro-informatique, à l'Ecole Nationale d'ingénieurs de Sfax (ENIS). Mention très bien.
- **1999-2002** • Diplôme d'ingénieur en Génie Electrique, option Micro-informatique, à l'Ecole Nationale d'ingénieurs de Sfax (ENIS). Mention bien.

Parcours Professionnel :

- **Depuis septembre 2009** • Maître de Conférences à l'Université de Valenciennes et du Hainaut Cambrésis (UVHC) et chercheur du Laboratoire d'Automatique, de Mécanique et d'Informatique industrielles et Humaines (LAMIH) UMR UVHC/CNRS 8201.
- **Octobre 2008-Aout 2009** • Post-doctorant en informatique à l'Université de Valenciennes, Laboratoire LAMIH, équipe SIADE dans le cadre du Projet ANR PrimaCare.
- **Mars 2008-Septembre 2008** • Post-doctorant en informatique à l'INRIA Lille-Nord Europe, EPI (Equipe Projet INRIA) DaRT dans le cadre du projet Ter@ops du pôle de compétitivité System@tic.
- **Novembre 2004-Février 2008** • Doctorant en informatique à l'USTL au sein de l'EPI DaRT (INRIA Lille-Nord Europe). Bourse dans le cadre du projet européen Interreg III A ModEasy.

A.2 Activités d'enseignement

A.2.1 Service réalisé 2009-2014

Les tableaux ci-dessous résument les activités d'enseignements réalisées entre 2009 et 2014 à l'Université de Valenciennes et du Hainaut-Cambrésis au sein de l'Institut des Sciences et Techniques de Valenciennes (ISTV), en qualité de maître de conférences en Informatique. Certain enseignements ont nécessité la préparation de l'intégralité des supports de cours, TD, TP et examen. Le volume horaire global pour chaque année de la période 2009 - 2014 est donné dans le Tableau A.1 et celui des modules enseignés est détaillé dans le Tableau A.2.

TABLE A.1 – Service réalisé pour la période 2009 - 2014 à l'ISTV

Année universitaire	Heures eq. TD
2009/2010	211
2010/2011	285
2011/2012	310
2012/2013	285
2013/2014	313

TABLE A.2 – Modules enseignés pour la période 2009 - 2014 à l'ISTV

Modules	Niveau	Cours	TD	TP	Heures eq. TD
Algorithmique et programmation langage C	L1	-	18	-	18
Introduction aux systèmes d'exploitation et aux architectures des ordinateurs	L2 Info	15	20	15	57,5
Développement des applications informatiques niveau 2	L2 Info	6	6	12	27
Structures de données avancées	L2 Info	12	12	12	42
Algorithmique et programmation impérative	L2 Info	15	20	15	57,5
Architecture des ordinateurs	L3 Info	-	12	-	12
OS réseaux et programmation des systèmes	M1 IRCOMS	12	12	-	30
Architecture avancée des systèmes informatiques	M1 TNSI	-	12	-	12
Introduction à l'informatique embarquée	M1 TNSI	-	12	-	12
Remise à niveau en système d'exploitation et architecture	M2 CCI	12	12	12	42
Conception et modélisation des systèmes embarqués	M2 TNSI	12	6	6	30
Compilation et développement des applications embarquées	M2 TNSI	12	6	6	30

A.2.2 Participation à des jurys et suivi de stages

Durant la période 209 - 2014, j'ai participé à différents jurys de soutenances sanctionnant stages (Licence ou Master). J'ai également participé aux jurys de session pour les promotions dans lesquelles je suis intervenu. De plus, j'ai assuré le suivi de stages de Licence 3 et Master 2 en formation initiale, continue et par apprentissage.

A.2.3 Autres activités et responsabilités pédagogiques et administratives

- Responsable pédagogique de la Licence 2 Informatique à l'Institut des Sciences et Techniques de Valenciennes (ISTV) depuis janvier 2010. Cette formation intègre 45 étudiants pour l'année 2014 - 2015. Afin d'assurer la qualité pédagogique des enseignements, les étudiants sont répartis en deux groupes de TD et 2 groupes de TP.
- Membre du comité de sélection de Maître de Conférences en Informatique à l'Université de Valenciennes et du Hainaut-Cambrésis

A.3 Activités de recherche

Au cours du dernier quadriennale, les travaux de recherche que j'ai menés s'inscrivent dans le contexte de conception des systèmes multiprocesseur sur puce (MPSoC) et des systèmes dynamiquement reconfigurables. Au niveau du flot de conception, mes travaux sont centrés autour de l'estimation des performances et de la consommation d'énergie dans une perspective d'exploration architecturale de haut niveau. Sur le plan matériel, mes contributions concernent le développement des architectures multiprocesseur homogènes et hétérogènes CPU/FPGA ainsi que des architectures massivement parallèles dynamiquement reconfigurables. Le domaine d'applications couvre le transport intelligent en collaboration avec Airbus Group, Airbus Helicopters, Nolam Embedded Systems, etc.

A.3.1 Evaluation PEDR session 2014

J'ai présenté ma demande de Prime d'Encadrement Doctoral et de Recherche (PEDR) pour l'année 2014 - 2015. Le dossier de candidature a été étudié par le Conseil National des Universités (CNU) section 27 et classé dans les 20% meilleurs parmi les dossiers examinés avec une note globale A. Les détails des éléments scientifiques d'évaluation sont donnés dans le tableau suivant :

Eléments scientifiques d'évaluation	De la plus grande qualité	Satisfait pleinement aux critères	Doit être consolidé en vue d'une prime	Insuffisamment renseigné
1 - Publications / production scientifique	X			
2 - Encadrement doctoral scientifique	X			
3 - Diffusion des travaux	X			
4 - Responsabilités scientifiques	X			

A.3.2 Publications et production scientifique

Brevets (3)

B1. Martial Rubio (Airbus Helicopters), Nicolas Belanger (Airbus Helicopters), Rabie Ben Atitallah (LAMIH/UVHC), and Jean-Luc Dekeyser (LIFL/Lille1). "Procédé d'optimisation dynamique d'une architecture d'outils de tests système". (2011) Brevet International enregistré à l'Institut National de la Propriété Industrielle sous le n° B64F5/00 ; G06F11/30 ; G06F15/16. Brevet accessible à partir du site http ://www.inpi.fr/

B2. Martial Rubio (Airbus Helicopters), Nicolas Belanger (Airbus Helicopters), Rabie Ben Atitallah (LAMIH/UVHC), and Jean-Luc Dekeyser (LIFL/Lille1). (2012) "A method of dynamically optimizing an architecture of system test tools". Brevet International enregistré à l'Australian Patent Office sous le n° 2012200402. Brevet accessible à partir du site http ://www.ipaustralia.gov.au

B3. George Afonso (Airbus IW), Wenceslas Godard (Airbus IW), Rabie Ben Atitallah (LAMIH/UVHC), and Jean-Luc Dekeyser (LIFL/Lille1). "Système de simulation et de test". (2014) Brevet International déposé à l'Institut National de la Propriété Industrielle sous le n°/Ref : 27092 FR. Brevet en cours de publication.

Revues internationales avec comité de lecture (7 publiées + 3 en cours de révision)

J1. AIT EL CADI A., BEN ATITALLAH R., HANAFI S., MLADENOVIC N., ARTIBA A. New MIP model for Multiprocessor Scheduling Problem with Communication Delays. Journal of Optimisation Letters, accepted in sep. 2014, Springer DOI : 10.1007/s11590-014-0802-2.

J2. BEN ATITALLAH R., SENN E., CHILLET D., LANOE M., BLOUIN D. An efficient Framework for Power-Aware Design of Heterogeneous MPSoC. IEEE Transactions on Industrial Informatics, Volume : 9, Issue : 1, Page(s) : 487 – 501, 2013.

J3. ABDALLAH A., GAMATIE A., **BEN ATITALLAH R.**, DEKEYSER J-L. Abstract Clock-based Design of a JPEG Encoder. IEEE Embedded Systems Letters, Volume : 4, Issue : 2 Page(s) : 29 - 32, 2012.

J4. Chtioui H., Niar S., **BEN ATITALLAH R.**, Zahran M., Dekeyser J-L., Abid M. A Dynamic Hybrid Cache Coherency Protocol for Shared-Memory MPSoC Architectures. International Journal of Computer Applications (IJCA), 11, pp. 1, ISSN 0975 – 8887, 2012.

J5. GAMATIE A., LEBEUX S., PIEL E., **BEN ATITALLAH R.**, ETIEN A., MARQUET P., DEKEYSER J-L. A model driven design framework for regular and massively parallel embedded applications. ACM Transactions in Embedded Computing Systems (TECS), volume 10, Issue 4, November 2011.

J6. BEN ATITALLAH R., PIEL E., NIAR S., MARQUET P., DEKEYSER J-L. A Fast MPSoC Virtual Prototyping for Intensive Signal Processing Applications. Microprocessors and Microsystems Embedded Hardware Design Journal (MICPRO), 36 issue 3, pp. 176–189, July 2011.

J7. TRABELSI C., **BEN ATITALLAH R.**, MEFTALI S., JEMAI A., DEKEYSER J-L. A Model-Driven Approach for Hybrid Power Estimation in Embedded Systems Design. EURASIP Journal on Embedded Systems, "2011", pp. "15", ISSN 10.1155/2011/569031, February 2011.

Revues en cours de révision

J8. BEN ATITALLAH R., VISWANATHAN V., DEKEYSER J-L. A Reconfigurable Technology-Centric Design Process for Avionic Simulation and Test. Submitted to IEEE Transactions on Industrial Informatics, August 2014.

J9. OUNI B., MHEDBI I., TRABELSI C., **BEN ATITALLAH R.**, BELLEUDY C., Multi-Level Energy/Power-Aware Design Methodology for MPSoC. Submitted to IEEE Transactions on Computer-Aided Design, August 2014.

J10. Ait El Cadi A., SOUISSI O., **BEN ATITALLAH R.**, ARTIBA A. Mathematical Programming Models for Scheduling in a CPU/FPGA Architecture with Heterogeneous Communication delays. Submitted to Journal of Intelligent Manufacturing (JIM), July 2014.

Conférences internationales avec actes et comité de lecture (33)

C1. ALI K., BEN ATITALLAH R., HANAFI S., DEKEYSER J-L. A Generic Pixel Distribution Architecture for Parallel Video Processing International Conference on ReConFigurable Computing and FPGAs (ReConFig 2014), Cancun, Mexico, December 2014.

C2. BEN ABDALLAH F., TRABELSI C., BEN ATITALLAH R., ABED M. Early Power-aware Design Space Exploration for Embedded Systems : MPEG-2 Case Study. International Symposium on System-on-Chip 2014, Tampere, Finland, October 28-29, 2014.

C3. TRABELSI C., BEN ATITALLAH R., MEFTALI S., DEKEYSER J-L. Model-Driven design flow for distributed control in reconfigurable FPGA systems. Conference on Design & Architectures for Signal & Image Processing (DASIP 2014), Madrid, Spain, October 2014.

C4. BOUAIN M., VISWANATHAN V., BEN ATITALLAH R., DEKEYSER J-L. Dynamic reconfiguration of modular I/O IP cores for avionic applications, 9th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC'2014), Montpellier, France, May 2014.

C5. VISWANATHAN V., BEN ATITALLAH R., DEKEYSER J-L. Redefining the role of FPGAs in the next generation avionic systems, 22nd ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA 2014), Monterey, California, February 2014.

C6. RETHINAGIRI S.K., PALOMAR O., UNSAL O., CRISTAL A., BEN ATITALLAH R., NIAR S. PETS : Power and energy estimation tool at system-level. 15th International Symposium on Quality Electronic Design (ISQED), 2014 (ISQED 2014), CA, Santa Carla, March 2014.

C7. BAKLOUTI Z., DUVIVIER D., BEN ATITALLAH R., ARTIBA A., BELANGER N. Real-time simulator supporting Heterogeneous CPU/FPGA architecture. International Conference on Industrial Engineering and Systems Management (IEEE IESM 2013), I4e2, Rabat, Morocco, October 2013.

C8. SOUISSI O., BEN ATITALLAH R., DUVIVIER D., ARTIBA A., BELANGER N., FEYZEAU P. Path Planning : A 2013 Survey. International Conference on Industrial Engineering and Systems Management (IEEE IESM 2013), I4e2, Rabat, Morocco, October 2013.

C9. AIT EL CADI A., BEN ATITALLAH R., ARTIBA A. Mathematical Programming Models for Scheduling in a CPU/FPGA Architecture with Communication Delay. International Conference on Industrial Engineering and Systems Management (IEEE IESM 2013), I4e2, Rabat, Morocco, October 2013.

C10. SOUISSI O., BEN ATITALLAH R., DUVIVIER D., ARTIBA A. Optimization Of Matching and Scheduling On Heterogeneous CPU/FPGA Architectures. 7th IFAC Confe-

rence on Manufacturing Modelling, Management, and Control, Saint Petersburg, Russia, June 2013.

C11. AFONSO G., BAKLOUTI Z., DUVIVIER D., BEN ATITALLAH R., BILLAUER E. HETEROGENEOUS CPU/FPGA RECONFIGURABLE COMPUTING SYSTEM FOR AVIONIC TEST APPLICATION. Reconfigurable Architectures Workshop (RAW 2013), Boston, USA, May 2013.

C12. AFONSO G., DAMIANI N., BELANGER N., BEN ATITALLAH R., RUBIO M. Hybrid and multicore optimized architectures for test and simulation systems. The 6th International ICST Conference on Simulation Tools and Techniques (SIMUTools 2013), Cannes, French Riviera, March 2013.

C13. VISWANATHAN V., BEN ATITALLAH R., NAKACHE B., NAKACHE M., DEKEYSER J-L. Dynamic reconfiguration of modular I/O IP cores for avionic applications. International Conference on ReConFigurable Computing and FPGAs (ReConFig 2012), Cancun, Mexico, December 2012.

C14. SENN E., BELLEUDY C., CHILLET D., FRITSCH A., BEN ATITALLAH R., ZENDRA O. "Open-People : Open Power and Energy Optimization PLatform and Estimator". 15th EUROMICRO Conference on Digital System Design (DSD'2012), Cesme, Izmir, Turkey, September 2012.

C15. SOUISSI O., BEN ATITALLAH R., ARTIBA A., ELMAGHRABY S. Optimization Of Run-time Mapping On Heterogeneous CPU/FPGA Architectures. 9th International Conference on Modeling, Optimization and SIMulation (MOSIM 2012), Bordeaux, France, June 2012.

C16. RETHINAGIRI S-K., BEN ATITALLAH R., SENN E., DEKEYSER J-L., NIAR S. An Efficient Power Estimation Methodology for Complex RISC Processor based Embedded Platforms. 22nd Great Lakes Symposium on VLSI (GLSVLSI 2012), Salt Lake City, Utah, USA, May 2012.

C17. MHEDBI I., BEN ATITALLAH R., JEMAI A. Dynamic Slack Reclamation Strategy for Multiprocessor Systems. The 16 IEEE Mediterranean Electrotechnical Conference, Hammamet, Tunisia, March 2012.

C18. AFONSO G., BEN ATITALLAH R., DEKEYSER J-L. Software Implementation vs. Hardware Implementation : The Avionic Test System Case-Study. Architectural Support for Programming Languages and Operating Systems (ASPLOS 2012), London, United Kingdom, March 2012.

C19. RETHINAGIRI S-K., BEN ATITALLAH R., SENN E., NIAR S., DEKEYSER J-L. Fast and Accurate Hybrid Power Estimation Methodology for Embedded Systems. Conference on Design & Architectures for Signal & Image Processing, Tampere FL, November 2011.

C20. RETHINAGIRI S-K., BEN ATITALLAH R., DEKEYSER J-L. A System Level Power Consumption Estimation for MPSoC. International Symposium on System-on-Chip 2011 (SOC 2011), Tampere, Finland, October 2011.

C21. RETHINAGIRI S-K., BEN ATITALLAH R., NIAR S., SENN E., DEKEYSER J-L. Hybrid System Level Power Consumption Estimation for FPGA-Based MPSoC. International Conference on Computer Design (ICCD'11), September 2011.

C22. AFONSO G., BEN ATITALLAH R., LOYER A., DEKEYSER J-L., BELANGER N., RUBIO M. A prototyping environment for high performance reconfigurable computing. 6th International Workshop on Reconfigurable and Communication-centric Systems-on-Chip, Montpellier, France, June 2011.

C23. AFONSO G., BEN ATITALLAH R., BELANGER N., RUBIO M., STILKERICH S., DEKEYSER J-L. Toward Generic and Adaptive Avionic Test Systems. NASA/ESA Conference on Adaptive Hardware and Systems, San Diego, California, USA, June 2011.

C24. HARB N., NIAR S., SAGHIR M., ELHILLALI Y., BEN ATITALLAH R. Dynamically Reconfigurable Architecture for a Driver Assistant System. IEEE Symposium on Application Specific Processors (SASP 2011), San Diego, California, USA, June 2011.

C25. LANGE T., HARB N., NIAR S., LIU H., BEN ATITALLAH R. (2010). An Improved Automotive Multiple Target Tracking System Design. 13th EUROMICRO Conference on Digital System Design DSD'2010, Lille France, September 2010.

C26. AFONSO G., BEN ATITALLAH R., BELANGER N., RUBIO M., DEKEYSER J-L. An Efficient Design Methodology for Hybrid Avionic Test Systems. 15th IEEE International Conference on Emerging Techonologies and Factory Automation, Bilbao, Spain, January 2010.

C27. CHTIOUI H., BEN ATITALLAH R., NIAR S., DEKEYSER J-L., ABID M. A Dynamic Hybrid Cache Coherency Protocol for Shared-Memory MPSoC. 12th EUROMICRO Conference on Digital System Design, University Of Patras, Greece, August 2009.

C28. DEKEYSER J-L., GAMATIE A., ETIEN A., BEN ATITALLAH R. Using the UML Profile for MARTE to MPSOC Co-design. First International Conference on Embedded Systems & Critical Applications, Tunisia, May 2008.

C29. BEN ATITALLAH R., NIAR S., DEKEYSER J-L. MPSoC Power Estimation Framework at Transaction Level Modeling. In The 19th IEEE International Conference on Microelectronics (ICM 2007), Cairo, Egypt, December 2007.

C30. BEN ATITALLAH R., PIEL E., NIAR S., MARQUET P., DEKEYSER J-L. Multilevel MPSoC simulation using an MDE approach". In IEEE International SoC Conference (SoCC 2007), Hsinchu, Taiwan, September 2007.

C31. BEN ATITALLAH R., NIAR S., MEFTALI S., DEKEYSER J-L. An MPSoC performance estimation framework using transaction level modeling. In The 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'07), Daegu, Korea, August 2007.

C32. BEN ATITALLAH R., BONDE L., NIAR S., MEFTALI S., DEKEYSER J-L. Dekeyser. Multilevel MPSoC performance evaluation using MDE approach. In International Symposium on System-on-Chip 2006 (SOC 2006), Tampere, Finland, November 2006.

C33. BEN ATITALLAH R., NIAR S., GREINER A., MEFTALI S., DEKEYSER J-L. Estimating energy consumption for an MPSoC architectural exploration. In Architecture of Computing Systems (ARCS'06), Frankfurt, Germany, March 2006.

Conférences et workshops nationaux avec actes et comité de lecture (...)

W1. Rethinagiri S.K., Palomar O., **BEN ATITALLAH R.**, Unsal O., Cristal A., Niar S. System-level power estimation tool for embedded processor based platforms. 6th Workshop on Rapid Simulation and Performance Evaluation : Methods and Tools (RAPIDO 2014) in conjuction with Hipeac conference 2014, January, Vienna, Austria.

W2. BOUAIN M., **BEN ATITALLAH R.**, MASMOUDI N., Dekeyser J-L. Design Space Exploration on Heterogeneous SoC : The H.264 encoder case-study. GdR SOC-SIP, Lyon, France, June 2013.

W3. SENN E., Belleudy C., Chillet D., Fritsch A., **BEN ATITALLAH R.**, Zendra, O. OpenPEOPLE : Open Power and Energy Optimization PLatform and Estimator. 14th Sophia-

Antipolis Microelectronics Forum SAME 2011, Nice, France, December 2011.

W4. AFONSO G., **BEN ATITALLAH R.**, DEKEYSER J-L. A Design Environment for Reconfigurable Computing Systems. GdR SOC-SIP, Lyon, France, June 2011.

W5. RETHINAGIRI S-K., **BEN ATITALLAH R.**, NIAR S., SENN E., DEKEYSER J-L. An Effective Approach for Power Consumption Modeling of Complex Processor. GdR SOC-SIP, Lyon, France, June 2011.

W6. LIU H., NIAR S., **BEN ATITALLAH R.** An efficient scalable MPSoC architecture for dynamic task distribution. PROGram for Research on Embedded Systems & Software, STW.ICT, Veldhoven, Nederland, November 2010.

W7. TRABELSI C., MEFTALI S., **BEN ATITALLAH R.**, JEMAI A., DEKEYSER J-L., NIAR S. An MDE Approach for Energy Consumption Estimation in MPSoC Design. 2nd Workshop on Rapid Simulation and Performance Evaluation : Methods and Tools (RAPIDO 2010) in conjuction with Hipeac conference 2010, January, Pisa, Italy.

Rapport de contrat de recherche (1)

R1. ABDALLAH A., GAMATIE A., **BEN ATITALLAH R.**, DEKEYSER J-L. Correct and Energy-Efficient Design of a Multimedia Application on SoCs. INRIA Research Report n. 7715, August 2011.

A.3.3 Encadrement doctoral et scientifique :

Thèses de doctorat (2 thèses soutenues + 5 en cours)

Thèses soutenues

1. Santhosh Kumar Rethinagiri, thèse soutenue le 14 mars 2013, « Une approche système pour l'estimation de la consommation de puissance des plateformes MPSoC », Université de Valenciennes et du Hainaut Cambrésis.

- Date de début : 01/12/2009 Date de fin : 14/03/2013
- Encadrement (%) : 50%
- Nom et % des Co-directeurs : Jean-Luc Dekeyser 25%, Smail Niar 25%

2. George Afonso, thèse soutenue le 02 juillet 2013, « Vers une nouvelle génération de systèmes de test et de simulation avionique dynamiquement reconfigurables », Université de Lille1.

- Date de début : 01/07/2010 Date de fin : 02/07/2013
- Encadrement (%) : 50%
- Nom et % des Co-directeurs : Jean-Luc Dekeyser 50%

Thèses en cours

3. Omar Souissi, « Système embarqué à haute performance pour le calcul de la trajectoire 3D », Université de Valenciennes et du Hainaut Cambrésis.

- Date de début : 01/10/2011 Date de fin : 30/09/2014
- Encadrement (%) : 35%
- Nom et % des Co-directeurs : Abdelhakim Artiba 35%, David Duvivier 30%

4. Venkatasubramanian Viswanathan, « Modèle de reconfiguration parallèle pour des architectures Multi-FPGA», Université de Valenciennes et du Hainaut Cambrésis.

- Date de début : 01/02/2012 Date de fin : 31/01/2015
- Encadrement (%) : 50%

– Nom et % des Co-directeurs : Jean-Luc Dekeyser 50%

5. Konstanca Nikolajevic, « Système décisionnel embarqué pour le pilotage d'un hélicoptère en situation d'autonomie d'urgence », Université de Valenciennes et du Hainaut Cambrésis.

– Date de début : 01/11/2012 Date de fin : 31/10/2015
– Encadrement (%) : 20%
– Nom et % des Co-directeurs : Abdelhakim Artiba 30%, David Duvivier 50%

6. Karim Mohamed Ali, « Système reconfigurable à haute performance pour la sécurité du transport ferroviaire », Université de Valenciennes et du Hainaut Cambrésis.

– Date de début : 01/10/2013 Date de fin : 30/09/2016
– Encadrement (%) : 50%
– Nom et % des Co-directeurs : Jean-Luc Dekeyser 25%, Said Hanafi 25%

7. Wissem Chouchene, « Modèle de reconfiguration pour les FPGA 3D », Université de Lille1.

– Date de début : 01/10/2013 Date de fin : 30/09/2016
– Encadrement (%) : 50%
– Nom et % des Co-directeurs : Jean-Luc Dekeyser 50%

Master recherche (3)

1. Zeineb Baklouti, « Supervision temps réel des systèmes hétérogènes CPU/FPGA », Université de Valenciennes et du Hainaut Cambrésis.

– Date de début : 01/04/2012 Date de fin : 31/12/2012
– Encadrement (%) : 50%
– Nom et % des Co-directeurs : David Duvivier 50%

2. Yomna Ben Jmaa, « Gestion dynamique tension/fréquence pour les MPSoC en vue de l'optimisation de la consommation d'énergie», Université de Valenciennes et du Hainaut Cambrésis.

– Date de début : 01/04/2012 Date de fin : 30/09/2012
– Encadrement (%) : 100%

3. Imen Mhedbi, « Exploration de politiques de gestion dynamique tension/fréquence pour les MPSoC », Université de Valenciennes et du Hainaut Cambrésis.

– Date de début : 01/03/2011 Date de fin : 30/09/2011
– Encadrement (%) : 100%

A.3.4 Diffusion scientifique :

Séminaires et Tutoriaux :

– Rabie Ben Atitallah and Jean-Luc Dekeyser. "Un modèle de fonctionnement SPMD à base d'IPS". Ecole d'hiver Francophone sur les Technologies de Conception des Systèmes embarqués Hétérogènes (FETCH 2012), Janvier Alpe D'Huez, France.
– Rabie Ben ATITALLAH, "Model homogenization for power estimation and design exploration". International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS 2011), Tutorial session, septembre 2011, Madrid, Spain.

Distinction :

Best paper award : à la 29th IEEE International Conference on Computer Design (ICCD 2011) pour le papier RETHINAGIRI S-K., **BEN ATITALLAH R.**, NIAR S., SENN E., DEKEYSER J-L. Hybrid System Level Power Consumption Estimation for FPGA-Based MPSoC, September 2011.

Brevet :

Diffusion de l'article « Eurocopter a déposé un brevet issu de recherches de l'université de Valenciennes » dans le quotidien La voix du Nord autour du brevet [B1]. Cette innovation a été présentée dans le quotidien Voix du nord le 06/03/2011 : http://www.lavoixdunord.fr/Locales/Valenciennes/actualite/Valenciennes/2011/03/06/article_eurocopter-a-depose-un-brevet-issu-de-re.shtml

Démonstrations :

D1. Démonstration de la plateforme Open Power and Energy Optimization Platform and Estimator (Open-PEOPLE) à Design, Automation & Test in Europe (DATE 2011), University Booth, Grenoble, France, March 2011.

D2. Démonstration de la plateforme Open-PEOPLE à 14th Sophia-Antipolis Microelectronics Forum SAME 2011, Nice, France, December, 2011.

Collaborations académiques et industrielles avec diffusions scientifiques :

Académiques :

- LIFL, Université de Lille 1 : encadrements, publications et organisation d'événements scientifiques dans le cadre des projets ANR OpenPeople, CIFRE EADS IW, CIFRE NOLAM Embedded Systems
- INRIA équipes CAIRN et TRIO : tutoriaux et publications dans le cadre du projet ANR OpenPeople
- LEAT, Université de Nice Sophia-Antipolis : encadrement de master recherche et publication dans le cadre du projet ANR OpenPeople
- LAB-STICC, Université de Bretagne-Sud : tutoriaux et publications dans le cadre du projet ANR OpenPeople
- IEMN, UVHC : publication dans le cadre du projet ANR PRIMA-CARE
- L'Université Technologique de Delft (TU Delft) : publication dans le cadre de collaboration avec le laboratoire Software Engineering Research Group.
- North Carolina State University : encadrement et publication, financement UVHC (professeur invité)
- New York University : publication, financement UVHC (professeur invité)
- Ecole Nationale d'Ingénieurs de Sfax (ENIS) : encadrement et publications dans le cadre du projet STIC INRIA
- Institut National des Sciences Appliquées et de Technologie (INSAT), Tunis : encadrement et publications dans le cadre du projet INRIA Euromed 3+3.

Industrielles :

- EADS IW (Airbus Group) (France et Allemagne) : encadrement, brevet et publications dans le cadre de la CIFRE EADS IW/UVHC/INRIA

- Airbus Helicopters : encadrement, publications et brevet dans le cadre des contrats de collaboration UVHC/Eurocopter
- Nolam Embedded Systems : encadrement et publications dans le cadre de la CIFRE UVHC/Nolam
- Thales Communication : publications et tutoriaux dans le cadre du projet ANR OpenPeople

Transfert technologique :

Projet de transfert technologique à travers la composante VALUTEC SA (Filiale de l'Université de Valenciennes et du Hainaut Cambrésis) pour la réalisation d'un environnement d'exploration de l'espace de solutions pour la génération d'un plan de vol optimal (DTA-NAV). Ce projet est financé par Airbus Helicopters pour une durée de huit mois à partir de mars 2014. Il vise la diffusion d'un outil logiciel développé au sein du laboratoire dans le cadre de projets de recherche avec un niveau de maturité élevé.

A.3.5 Responsabilités scientifiques :

Activités scientifiques :

- Membre associé à l'équipe DaRT de l'INRIA Lille Nord Europe depuis octobre 2008 jusqu'à décembre 2012
- Participation au montage de la nouvelle équipe DreamPal de l'INRIA Lille Nord Europe
- Membre associé à la nouvelle équipe DreamPal de l'INRIA Lille Nord Europe depuis janvier 2013
- Membre du comité de pilotage du Groupe de Recherche (GdR) ASR (Architecture, Système et Réseau), responsable de l'action transverse autour de l'économie d'énergie dans les systèmes informatique
- Contributeur aux activités de recherche de l'IRT (Institut de Recherche Technologique) Railenium
- Membre du réseau d'excellence européen HiPEAC (High Performance and Embedded Architecture and Compilation) depuis 2007
- Membre du GdR SoC-SIP

Organisation de congrès

- Membre du comité d'organisation des journées francophones Green Days@Rennes (1-2 juillet 2014). Ces journées francophones sont organisées dans le cadre de l'action d'envergure Inria Hemera et de l'action transversale Energie et le Pôle Système du GDR ASR.
- Membre du comité d'organisation du workshop RAPIDO 2010/2011 en conjonction avec la conférence HiPEAC.
- Membre du comité d'organisation de Euromicro DSD et SEAA 2010 organisées à Lille en septembre 2010.

Responsabilité de contrats industriels ou publics

- Montage et pilotage du projet ANR-ARPEGE OpenPeople (2008-2012) au niveau de l'INRIA Lille Nord Europe (thèse Santhosh Kumar Rethinagiri) (Financement : 186 K€)
- Montage et pilotage de la CIFRE EADS IW/UVHC/INRIA (2010-2013) au niveau de l'Université de Valenciennes et du Hainaut Cambrésis (thèse George Afonso) (Financement : 30 K€)
- Montage et pilotage de la CIFRE UVHC/Nolam Embedded Systems (2011-2015) au niveau de l'Université de Valenciennes et du Hainaut Cambrésis (thèse Venkatasubramanian Viswanathan) (Financement : 10 K€)
- Montage et pilotage du contrat de collaboration UVHC/Airbus Helicopters (2011-2015) au niveau de l'Université de Valenciennes et du Hainaut Cambrésis (thèse Omar Souissi) (Financement : 70 K€)
- Montage de la CIFRE UVHC/Airbus Helicopters (2012-2015) (thèse Konstanca Nikolajevic) (Financement : 70 K€)
- Montage et pilotage du contrat post-doctorant (Abdessamad Ait El Kadi) dans le cadre de l'IRT Railenium (2013-2014) (Financement : 110 K€)
- Montage et pilotage du projet de transfert technologique VALUTEC-UVHC/Airbus Helicopters (2014) (Financement : 28 K€)

Comités de programme de conférences :

- The 6th IEEE International Workshop on Multicore and Multithreaded Architectures and Algorithms (M2A2) 2013 et 2014.
- Conference on Design & Architectures for Signal & Image Processing DASIP 2011 et 2012
- Workshop on Rapid Simulation and Performance Evaluation : Methods and Tools RAPIDO 2011, Crete, Greece.

Relecture de papiers de revues et de conférences :

- ACM Transactions on Design Automation of Electronic Systems, Journal of Real-Time Image Processing
- Conférences : DSD 2009/2010 ; ReCoSOC 2009/2010/2011 ; SoC 2009/2010/2011 ; DASIP 2011/2012 ; M2A2 2013/2014.

Evaluation de projet international :

- Evaluation d'un projet international « Flight Trajectory Optimization » pour le Conseil de recherches en sciences naturelles et en génie du Canada, Mai 2013.

Personal Bibliography

[1] A. Abdallah, A. Gamatié, R. Ben Atitallah, and J.-L. Dekeyser. Abstract clock-based design of a jpeg encoder. *Embedded Systems Letters*, 4(2):29–32, 2012.

[2] G. Afonso, Z. Baklouti, D. Duvivier, R. Ben Atitallah, E. Billauer, and S. Stilkerich. Heterogeneous cpu/fpga reconfigurable computing system for avionic test application. In *In 20th Reconfigurable Architectures Workshop*, BOSTON, USA, May 2013.

[3] G. Afonso, R. Ben Atitallah, N. Bélanger, M. Rubio, and J.-L. Dekeyser. An efficient design methodology for hybrid avionic test systems. In *15th IEEE International Conference on Emerging Techonologies and Factory Automation (ETFA 2010)*, Bilbao, Spain, September 2010.

[4] G. Afonso, R. Ben Atitallah, N. Bélanger, M. Rubio, S. Stilkerich, and J.-L. Dekeyser. Toward generic and adaptive avionic test systems. In *NASA/ESA Conference on Adaptive Hardware and Systems (AHS 2011)*, San Diego, California, USA, June 2011.

[5] G. Afonso, R. Ben Atitallah, and J.-L. Dekeyser. Software implementation vs. hardware implementation: The avionic test system case-study. In *Architectural Support for Programming Languages and Operating Systems (ASPLOS 2012)*, London, United Kingdom, March 2012.

[6] G. Afonso, R. Ben Atitallah, J.-L. Dekeyser, N. Belanger, and M. Rubio. A prototyping environment for high performance reconfigurable computing. In *6th International Workshop on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC'2011)*, Montpellier, France, June 2011.

[7] G. Afonso, N. Damiani, N. Belanger, R. Ben Atitallah, and M. Rubio. Hybrid and multicore optimized architectures for test and simulation systems. In *Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques*, SimuTools '13, Cannes, French Riviera, June 2013.

[8] G. Afonso, W. Godard, R. Ben Atitallah, and J.-L. Dekeyser. Système de simulation et de test. *Institut National de la Propriété Industrielle, Patent*, (Number: 27092 FR), 2014.

[9] A. Ait El Cadi, R. Ben Atitallah, and A. Artiba. Mathematical Programming Models for Scheduling in a CPU/FPGA Architecture with Communication Delay. In *International Conference on Industrial Engineering and Systems Management - IESM'2013*, Rabat, Maroc, October 2013.

[10] A. Ait El Cadi, R. Ben Atitallah, S. Hanafi, N. Mladenovic, and A. Artiba. New mip model for multiprocessor scheduling problem with communication delays. *Optimisation Letters*, September 2014, Springer DOI: 10.1007/s11590-014-0802-2, 2014.

[11] A. Ait El Cadi, O. Souissi, R. Ben Atitallah, and A. Artiba. Mathematical programming models for scheduling in a cpu/fpga architecture with heterogeneous communication delays. *Submitted to Journal of Intelligent Manufacturing*, July 2014.

[12] K. Ali, R. Ben Atitallah, J.-L. Dekeyser, and S. Hanafi. A generic pixel distribution architecture for parallel video processing. In *2014 International Conference on Reconfigurable Computing and FPGAs (ReConFig)*, Cancun, Mexico, December 2014.

[13] Z. Baklouti, D. Duvivier, R. Ben Atitallah, A. Artiba, and N. Belanger. Real-time simulator supporting heterogeneous cpu/fpga architecture. In *International Conference on Industrial Engineering and Systems Management - IESM'2013*, Rabat, Maroc, October 2013.

[14] R. Ben Atitallah, S. Niar, and J.-L. Dekeyser. MPSoC Power Estimation Framework at Transaction Level Modeling. In *The 19th International Conference on Microelectronics (ICM 2007)*, Cairo, Egypt, December 2007.

[15] R. Ben Atitallah, É. Piel, S. Niar, P. Marquet, and J.-L. Dekeyser. A fast mpsoc virtual prototyping for intensive signal processing applications. *Microprocessors and Microsystems - Embedded Hardware Design*, 36(3):176–189, 2012.

[16] R. Ben Atitallah, E. Senn, D. Chillet, M. Lanoe, and D. Blouin. An efficient framework for power-aware design of heterogeneous mpsoc. *IEEE Trans. Industrial Informatics*, 9(1):487–501, 2013.

[17] R. Ben Atitallah, V. Viswanathan, N. Belanger, and J.-L. Dekeyser. A reconfigurable technology-centric design process for avionic simulation and test. *Submitted to IEEE Trans. Industrial Informatics*, August 2014.

[18] M. Bouain, V. Viswanathan, R. Ben Atitallah, and J.-L. Dekeyser. Communication-centric design for fmc based i/o system. In *9th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC'2014)*, Montpellier, France, May 2014.

[19] H. Chtioui, R. Ben Atitallah, S. Niar, J. Dekeyser, and M. Abid. A dynamic hybrid cache coherency protocol for shared-memory mpsoc. In *12th Euromicro Conference on Digital System Design, Architectures, Methods and Tools, DSD'09.*, Patras, Greece, August 2009.

[20] H. Chtioui, S. Niar, R. Ben Atitallah, M. Zahran, J.-L. Dekeyser, and M. Abid. A dynamic hybrid cache coherency protocol for shared-memory mpsoc architectures. *International Journal of Computer Applications*, 47(3):45–50, June 2012.

[21] A. Gamatié, S. L. Beux, É. Piel, R. Ben Atitallah, A. Etien, P. Marquet, and J.-L. Dekeyser. A model-driven design framework for massively parallel embedded systems. *ACM Trans. Embedded Comput. Syst.*, 10(4):39, 2011.

[22] N. Harb, S. Niar, M. Saghir, Y. Hillali, and R. Ben Atitallah. Dynamically reconfigurable architecture for a driver assistant system. In *IEEE 9th Symposium on Application Specific Processors (SASP)*, San Diego , CA, USA, June 2011.

[23] T. Lange, N. Harb, H. Liu, S. Niar, and R. Ben Atitallah. An improved automotive multiple target tracking system design. In *13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools (DSD 2010)*, Lille, France, September 2010.

[24] I. Mhedhbi, R. Ben Atitallah, and A. Jemai. Dynamic slack reclamation strategy for multiprocessor systems. In *MELECON - IEEE Mediterranean Electrotechnical Conference*, Hammamet, Tunisia, March 2012.

[25] B. Neji, Y. Aydi, R. Ben Atitallah, S. Meftaly, M. Abid, and J.-L. Dykeyser. Multistage Interconnection Network for MPSoC: Performances study and prototyping on FPGA. In *3rd International Design and Test Workshop, IDT 2008.*, Monastir, Tunisia, December 2008.

[26] B. Ouni, I. Mhedbi, C. Trabelsi, R. Ben Atitallah, and C. Belleudy. Multi-Level Energy/Power-Aware Design Methodology for MPSoC. *Submitted to IEEE Trans. Computer Aided Design*, August 2014.

[27] S. K. Rethinagiri, R. B. Atitallah, S. Niar, E. Senn, and J. Dekeyser. Fast and accurate hybrid power estimation methodology for embedded systems. In *2011 Conference on Design and Architectures for Signal and Image Processing, DASIP 2011*, Tampere, Finland, November 2011.

[28] S. K. Rethinagiri, R. Ben Atitallah, J.-L. Dekeyser, E. Senn, and S. Niar. An efficient power estimation methodology for complex risc processor-based platforms. In *ACM Great Lakes Symposium on VLSI*, Salt Lake City, Utah, USA, October 2012.

[29] S. K. Rethinagiri, R. Ben Atitallah, S. Niar, E. Senn, and J. Dekeyser. Hybrid system level power consumption estimation for fpga-based mpsoc. In *2011 IEEE 29th International Conference on Computer Design (ICCD)*, pages 239–246, Salt Lake City, Utah, USA, October 2011.

[30] S. K. Rethinagiri, O. Palomar, O. S. Unsal, A. Cristal, R. Ben Atitallah, and S. Niar. PETS: Power and energy estimation tool at system-level. In *International Symposium on Quality Electronic Design (ISQED 2014)*, Santa Clara, CA, USA, 2014.

[31] M. Rubio, N. Belanger, R. Ben Atitallah, and J.-L. Dekeyser. Procédé d'optimisation dynamique d'une architecture d'outils de tests système. *Institut National de la Propriété Industrielle, Patent FR 2970792 (B1), EP 2482193 (A1)*, (Number: FR20110000232 20110126), 2011.

[32] M. Rubio, N. Belanger, R. Ben Atitallah, and J.-L. Dekeyser. A method of dynamically optimizing an architecture of system test tools. *Australian Patent Office, Patent G06G7/72 (2006.01)*, (Number: 2012200402), 2012.

[33] E. Senn, D. Chillet, O. Zendra, C. Belleudy, R. Ben Atitallah, A. Fritsch, and C. Samoyeau. Open-people: An open platform for estimation and optimizations of energy consumption. In *DASIP*, Karlsruhe, Germany, October 2012.

[34] E. Senn, D. Chillet, O. Zendra, C. Belleudy, S. Bilavarn, R. Ben Atitallah, C. Samoyeau, and A. Fritsch. Open-people: Open power and energy optimization platform and estimator. In *DSD*, Cesme, Izmir, Turkey, September 2012.

[35] O. Souissi, R. Ben Atitallah, and A. Artiba. Optimization of matching and scheduling on heterogeneous cpu/fpga architectures. In *7th IFAC Conference on Manufacturing Modelling, Management, and Control (IFAC MIM 2013)*, Saint Petersburg, Russia, June 2013.

[36] O. Souissi, R. Ben Atitallah, A. Artiba, and S. E. Elmaghraby. Optimization of run-time mapping on heterogeneous cpu/fpga architectures. In *9th International Conference of Modeling, Optimization and Simulation - MOSIM'12*, Bordeaux, France, March 2012.

[37] O. Souissi, R. Ben Atitallah, D. Duvivier, A. Artiba, and N. Belanger. Path Planning: A 2013 Survey. In *International Conference on Industrial Engineering and Systems Management - IESM'2013*, Rabat, Maroc, October 2013.

[38] C. Trabelsi, R. B. Atitallah, S. Meftali, and J. Dekeyser. Model-driven design flow for distributed control in reconfigurable FPGA systems. In *2014 Conference on Design and Architectures for Signal and Image Processing, DASIP 2014*, Madrid, Spain, October 2014.

[39] C. Trabelsi, R. Ben Atitallah, S. Meftali, J.-L. Dekeyser, and A. Jemai. A model-driven approach for hybrid power estimation in embedded systems design. *EURASIP J. Emb. Sys.*, DOI:10.1155/2011/569031, 2011.

[40] V. Viswanathan, R. Ben Atitallah, J.-L. Dekeyser, B. Nakache, and M. Nakache. Dynamic reconfiguration of modular i/o ip cores for avionic applications. In *2012 International Conference on Reconfigurable Computing and FPGAs (ReConFig)*, Cancun, Mexico, December 2012.

[41] V. Viswanathan, R. Ben Atitallah, J.-L. Dekeyser, B. Nakache, and M. Nakache. Redefining the role of fpgas in the next generation avionic systems. In *Proceedings of the 2014 ACM/SIGDA International Symposium on Field-programmable Gate Arrays*, FPGA '14, pages 248–248, New York, NY, USA, 2014. ACM.

References

[42] Pico computing hpc solutions. http://picocomputing.com/products/hpc-systems/. Online : Pico Computing.

[43] The SAE AADL Standard Info Site.

[44] SPICE manual. University of Berkeley (USA), URL : http://bwrc.eecs.berkeley.edu/Classes/IcBook/SPICE/.

[45] Storm simulation tool, 2011.

[46] C. Ababei, Y. Feng, B. Goplen, H. Mogal, T. Zhang, K. Bazargan, and S. Sapatnekar. Placement and routing in 3d integrated circuits. *IEEE Des. Test*, 22(6) :520–531, Nov. 2005.

[47] N. Abdelli, A. Fouilliart, N. Mien, and E. Senn. High-Level Power Estimation of FPGA. In *Industrial Electronics, 2007. ISIE 2007. IEEE International Symposium on*, pages 925–930. IEEE, 2007.

[48] A. Acquaviva, L. Benini, and B. Ricco. Energy characterization of embedded real-time operating systems. In *Proceedings of the Workshop on Compilers and Operating Systems for Low Power (COLP01)*, September 2001.

[49] G. Afonso. *Vers une nouvelle génération de systèmes de test et de simulation avionique dynamiquement reconfigurables*. PhD thesis, Université de Lille1, Laboratoire d'Informatique Fondamentale de Lille, Universitéde Lille 1, July 2013. v

[50] G. Afonso, N. Damiani, N. Belanger, R. B. Atitallah, and M. Rubio. Hybrid and multicore optimized architectures for test and simulation systems. In *Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques*, SimuTools '13, Cannes, French Riviera, 2013.

[51] T. A. AlEnawy and H. Aydin. Energy-aware task allocation for rate monotonic scheduling. In *procs. of IEEE Symposium, RTAS-2005*, pages 213–223, 2005.

[52] M. Anis and Y. Massoud. Power Design Challenges in Deep-Submicron Technology. In *IEEE 46th Midwest Symposium on Circuits and Systems*, Cairo, Egypt, 2003. 3

[53] T. Arpinen, E. Salminen, T. D. Hamalainen, and M. Hannikainen. Extension to marte profile for modeling dynamic power management of embedded systems. In *W6 1st Workshop on Model Based Engineering for Embedded Systems Design (M-BED 2010)*, 2010.

[54] S. Asano, T. Maruyama, and Y. Yamaguchi. Performance Comparison of FPGA, GPU AND CPU in Image Processing. In *19th IEEE International Conference on Field Programmable Logic and Applications, FPL*, Prague, Czech Republic, Aug. 2009.

[55] R. B. Atitallah, S. Niar, , and J.-L. Dekeyser. Mpsoc power estimation framework at transaction level modeling. In *The 19th International Conference on Microelectronics (ICM 2007)*, 2007.

[56] K. Baynes, C. Collins, E. Fiterman, B. Ganesh, P. Kohout, C. Smit, T. Zhang, and B. Jacob. The performance and energy consumption of three embedded real-time operating systems. In *Proceedings of CASES01*, Atlanta, Georgia, USA, November 2001.

[57] K. Baynes, C. Collins, E. Fiterman, B. Ganesh, P. Kohout, C. Smit, T. Zhang, and B. Jacob. The performance and energy consumption of embedded real-time operating systems. *IEEE Transactions on Computers*, 11(52) :1454–1469, November 2003.

[58] N. Belanger, J. Bovier, J.-F. Gilot, J.-P. Lebailly, and M. Rubio. Multi-Core computers and PCI Express The future of data acquisition and control systems. In *ETTC International Conference*, Toulouse, France, 2009.

[59] N. Belanger, N. Favarcq, and Y. Fusero. An open real time test system approach. In *IEEE International Confererence on Advances in System Testing and Validation Lifecycle*, Porto, Portugal, Sept. 2009.

[60] N. Belanger and J.-P. Lebailly. Promoting avionic test systems as productivity enablers. In *The fifth International Conference on Systems*, Les menuires, France, 2010.

[61] G. Beltrame, L. Fossati, and D. Sciuto. ReSP : A Nonintrusive Transaction-Level Reflective MPSoC Simulation Platform for Design Space Exploration. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 28(12) :1857–1869, Dec. 2009.

[62] A. D. M. G. Benini, L. ; Bogliolo. System-level dynamic power management. In *Low-Power Design, 1999. Proceedings. IEEE Alessandro Volta Memorial Workshop on*, pages 23 – 31, 2002.

[63] A. P. G. D. M. G. Benini, L. ; Bogliolo. Policy optimization for dynamic power management . *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 18(6) :1857–1869, 2002.

[64] L. Benini, E. Flamand, D. Fuin, and D. Melpignano. P2012 : Building an ecosystem for a scalable, modular and high-efficiency embedded computing accelerator. In *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '12, pages 983–987, San Jose, CA, USA, 2012. EDA Consortium. 4, 5

[65] P. Benoit, G. Sassatelli, P. Maurine, L. Torres, N. Azemard, M. Robert, F. Clermidy, M. Belleville, D. Puschini, B. Rebaud, O. Brousse, and M. Almeida. Towards autonomous scalable integrated systems. In G. Nicolescu, Ian O Connor, C. Piguet, editor, *Design Technology for Heterogeneous Embedded Systems*, pages 63–89. Springer, Mar. 2012.

[66] M. Berekovic, A. Kanstein, B. Mei, and B. D. Sutter. Mapping of nomadic multimedia applications on the ADRES reconfigurable array processor. *Microprocessors and Microsystems - Embedded Hardware Design*, 33(4) :290–294, 2009.

[67] C. A. M. L. R. L. U. o. N.-S. A. N. F. Bhatti, M.K. ; Belleudy. An inter-task real time dvfs scheme for multiprocessor embedded systems. In *Design and Architectures for Signal and Image Processing (DASIP), 2010 Conference on*, pages 136–143, 2011.

[68] K. Bhatti. *Energy-aware Scheduling for Multiprocessor Real-time Systems*. PhD thesis, Université de Nice Sophia-Antipolis, 2011.

[69] D. G. Bobrow, R. P. Gabriel, and J. L. White. Object-oriented programming. chapter CLOS in Context : The Shape of the Design Space, pages 29–61. MIT Press, Cambridge, MA, USA, 1993.

[70] D. Brooks, V. Tiwari, and M. Martonosi. Wattch : a framework for architectural-level power analysis and optimizations. In *Proceedings of the 27th annual international symposium on Computer architecture*, pages 83–94, 2000.

[71] L. A. Cardona, J. Agrawal, Y. Guo, J. Oliver, and C. Ferrer. Performance-area improvement by partial reconfiguration for an aerospace remote sensing application. In *Reconfigurable Computing and FPGAs (ReConFig), 2011 International Conference on*, 2011. 7, 9

[72] A. Cevrero, P. Athanasopoulos, H. Parandeh-Afshar, P. Brisk, Y. Lebebici, P. Ienne, and M. Skerlj. 3d configuration caching for 2d fpgas. In *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, FPGA '09, pages 286–286, New York, NY, USA, 2009. ACM.

[73] C.-I. Chen, B.-C. Lee, and J.-D. Huang. Architectural exploration of 3d fpgas towards a better balance between area and delay. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2011*, pages 1–4, March 2011.

[74] Y. Chen. *The Analysis and Practice on Open Source Embedded System Software–Based on SkyEye and ARM Developing Platform*. Beihang University Press, 2004.

[75] Z. Chen, R. N. Pittman, and A. Forin. Combining multicore and reconfigurable instruction set extensions. In *Proceedings of the 18th Annual ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, FPGA '10, pages 33–36, New York, NY, USA, 2010. ACM.

[76] F. Cloute, J.-N. Contensou, D. Esteve, P. Pampagnin, P. Pons, and Y. Favard. Hardware/software co-design of an avionics communication protocol interface system : an industrial case study. In *Hardware/Software Codesign, 1999. (CODES '99) Proceedings of the Seventh International Workshop on*, 1999.

[77] P. Coussy, C. Chavet, P. Bomel, D. Heller, E. Senn, and E. Martin. *GAUT : A High-Level Synthesis Tool for DSP Applications*. Springer Netherlands, 2008.

[78] M. F. da S. Oliveira, E. W. Brião, F. A. Nascimento, and F. R. Wagner. Model driven engineering for mpsoc design space exploration. In *SBCCI'07 :Proceedings of the 20th annual conference on Integrated circuits and systems design*, Rio de Janeiro, Brazil, 2007.

[79] M. F. da S. Oliveira, L. B. de Brisolara, L. Carro, and F. R. Wagner. Early embedded software design space exploration using uml-based estimation. In *Rapid System Prototyping*, 2006.

[80] N. Dhanwada, R. A. Bergamaschi, W. W. Dungan, I. Nair, P. Gramann, W. E. Dougherty, and I.-C. Lin. Transaction-level modeling for architectural and power analysis of powerpc and coreconnect-based systems.

[81] S. Dhouib, J.-P. Diguet, D. Blouin, and J. Laurent. Energy and power consumption estimation for embedded applications and operating systems. *Journal of Low Power Electronics (JOLPE)*, 5(3), 2009.

[82] C. Dong, D. Chen, S. Tanachutiwat, and W. Wang. Performance and power evaluation of a 3d cmos/nanomaterial reconfigurable architecture. In *Computer-Aided Design, 2007. ICCAD 2007. IEEE/ACM International Conference on*, pages 758–764, Nov 2007.

[83] M. Duranton, D. Black-Schaffer, K. De Bosschere, and J. Maebe. *THE HIPEAC VISION FOR ADVANCED COMPUTING IN HORIZON 2020*. HiPEAC network of excellence, 2013. 3, 4

[84] D. Elléouet, N. Julien, and D. Houzet. A high level soc power estimation based on ip modeling. In *Proceedings of the 20th international conference on Parallel and distributed processing*, IPDPS'06, pages 208–208, Washington, DC, USA, 2006. IEEE Computer Society.

[85] D. Elleouet, Y. Savary, and N. Julien. A fpga power aware design flow. In *Power and Timing Modeling, Optimization and Simulation, PATMOS*, pages 415–424, Montpellier, France, September 2006.

[86] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger. Dark silicon and the end of multicore scaling. *SIGARCH Comput. Archit. News*, 39(3) :365–376, June 2011. 5

[87] S. Fuller and L. Millett. Computing performance : Game over or next level ? *Computer*, 44(1) :31–38, Jan 2011. 3, 4

[88] K. Funaoka, S. Kato, and N. Yamasaki. Energy-efficient optimal real-time scheduling on multiprocessors. *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, 0 :23–30, 2008.

[89] A. D. G. G, J. L. Danger, and W. P. Burleson. Reducing the power consumption in fpgas with keeping a high performance level. *VLSI, IEEE Computer Society Workshop on*, 0 :47, 2000.

[90] A. Garcia, W. Burleson, and J. Danger. Power modelling in field programmable gate arrays (FPGA). *Lecture notes in computer science*, pages 396–404, 1999.

[91] A. Garcia, W. Burleson, and J. Danger. Power modelling in field programmable gate arrays (FPGA). In *Field Programmable Logic and Applications*, pages 396–404. Springer, 2004.

[92] GAUT : High-Level Synthesis tool From C to RTL . http://www-labsticc.univ-ubs.fr/www-gaut/.

[93] A. Gayasen, V. Narayanan, M. Kandemir, and A. Rahman. Designing a 3-d fpga : Switch box architecture and thermal issues. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 16(7) :882–893, July 2008.

[94] D. Gohringer, M. Hubner, V. Schatz, and J. Becker. Runtime adaptive multi-processor system-on-chip : Rampsoc. In *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, April 2008.

[95] M. Gries and K. Keutzer. *Building ASIPs : The Mescal Methodology*. Springer, 2005.

[96] A. Jacobs, G. Cieslewski, A. D. George, A. Gordon-Ross, and H. Lam. Reconfigurable fault tolerance : A comprehensive framework for reliable and adaptive fpga-based space computing. *ACM Trans. Reconfigurable Technol. Syst.*, 5(4) :21 :1–21 :30, Dec. 2012.

[97] A. B. Kahng. The itrs design technology and system drivers roadmap : Process and status. In *Proceedings of the 50th Annual Design Automation Conference*, DAC '13, pages 34 :1–34 :6, New York, NY, USA, 2013. ACM. 3

[98] R. S. Kihwan Choi, Wonbok Lee and L. A.-C. Massoud Pedram Department of EE-Systems, University of Southern California. Dynamic voltage and frequency scaling

under a precise energy model considering variable and fixed components of the system power dissipation.

[99] H. Krichene, M. Baklouti, M. Abid, P. Marquet, and J. Dekeyser. Broadcast with mask on a massively parallel processing on a chip. In *2012 International Conference on High Performance Computing & Simulation, HPCS 2012, Madrid, Spain, July 2-6, 2012*, 2012.

[100] S. Kumar Rethinagiri. *System-level power estimation methodology for MPSoC based platforms*. PhD thesis, Université de Valenciennes, Laboratoire d'Automatique, de Mécanique et d'Informatique industrielles et Humaines, March 2013. v

[101] M. Lanuzza, P. Zicari, F. Frustaci, S. Perri, and P. Corsonello. Exploiting self-reconfiguration capability to improve sram-based fpga robustness in space and avionics applications. *ACM Trans. Reconfigurable Technol. Syst.*, 4(1), dec 2010.

[102] J. Laurent, N. Julien, and E. Martin. Softexplorer : estimation, characterization and optimization of the power and energy consumption at the algorithmic level. In *Fourteenth International Workshop on Power and Timing Modeling (PATMOS 2004)*, pages 15–17, Santorini, Greece, September 2004.

[103] J. Laurent, N. Julien, E. Senn, and E. Martin. Functional Level Power Analysis : An efficient approach for modeling the power consumption of complex processors. In *Proc. Design Automation and Test in Europe DATE*, Paris, France, march 2004.

[104] I. Lee, H. Kim, P. Yang, S. Yoo, E. Chung, K.Choi, J.Kong, and S.Eo. Powervip : Soc power estimation framework at transaction level. In *Proc. ASP-DAC*, 2006.

[105] M. Lin, A. El Gamal, Y.-C. Lu, and S. Wong. Performance benefits of monolithically stacked 3d-fpga. In *Proceedings of the 2006 ACM/SIGDA 14th International Symposium on Field Programmable Gate Arrays*, FPGA '06, pages 113–122, New York, NY, USA, 2006. ACM.

[106] M. Lin, A. El Gamal, Y.-C. Lu, and S. Wong. Performance benefits of monolithically stacked 3d-fpga. In *Proceedings of the 2006 ACM/SIGDA 14th International Symposium on Field Programmable Gate Arrays*, FPGA '06, pages 113–122, New York, NY, USA, 2006. ACM.

[107] F. Liu, F. Guo, Y. Solihin, S. Kim, and A. Eker. Characterizing and modeling the behavior of context switch misses. In *International Conference on Parallel Architectures and Compilation Techniques*, pages 91–101, 2008.

[108] P. Marquet, S. Duquennoy, S. Le Beux, S. Meftali, and J.-L. Dekeyser. Massively parallel processing on a chip. In *Proceedings of the 4th International Conference on Computing Frontiers*, CF '07, pages 277–286, New York, NY, USA, 2007. ACM.

[109] N.Dhanwada, I. Lin, and V.Narayanan. A power estimation methodology for systemc transaction level models. In *International conference on Hardware/software codesign and system synthesis*, 2005.

[110] F. Nicolas, F. Antoine, and F. Paul. esimu : a fast and accurate energy consumption simulator for real embedded system. In *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM 2007*, pages 1–6, June 2007.

[111] OMG. Uml profile for schedulability, performance, and time, 2002.

[112] OMG. The official omg marte web site. http://www.omgmarte.org, May 2007.

[113] Open SystemC Initiative. Systemc, 2008. World Wide Web document, URL : http://www.systemc.org/.

[114] Opencore. Jpeg encoder, 2013.

[115] B. Osterloh, H. Michalik, S. Habinc, and B. Fiethe. Dynamic partial reconfiguration in space applications. In *Adaptive Hardware and Systems, 2009. AHS 2009. NASA/ESA Conference on*, 2009. 7, 9

[116] J. Park, D. Shin, N. Chang, and M. Pedram. Accurate modeling and calculation of delay and energy overheads of dynamic voltage scaling in modern high-performance microprocessors. In *Proceedings of the 16th ACM/IEEE international symposium on Low power electronics and design*, pages 419–424. ACM, 2010.

[117] K. D. Pham, A. Jain, J. Cui, S. Fahmy, and D. Maskell. Microkernel hypervisor for a hybrid arm-fpga platform. In *Application-Specific Systems, Architectures and Processors (ASAP), 2013 IEEE 24th International Conference on*, pages 219–226, June 2013.

[118] Philips Electronic Design and Tools Group. DIESEL User Manual. Technical report, Philips Research, June 2001.

[119] Planet MDE. *Model Driven Engineering*. http://planetmde.org/, 2007.

[120] H. Plankl. Embedded solutions for development tests, component tests and system integration in the test centre. In *Aerospace Testing*, Munich, Germany, 2009.

[121] S. project. An open platform for modelling and simulation of multi-processors system on chip. http://soclib.lip6.fr/Home.html.

[122] M. Reserach. emips : a dynamically extensible processor, 2006-2012.

[123] S. Rethinagiri, R. Ben Atitallah, S. Niar, E. Senn, and J. Dekeyser. Hybrid system level power consumption estimation for fpga-based mpsoc. In *Computer Design (ICCD), 2011 IEEE 29th International Conference on*, pages 239–246, Oct 2011.

[124] J. D. S. Douhib. Model driven high-level power estimation of embedded operating systems communication and synchronization services. In *Proceedings of the 6th IEEE International Conference on Embedded Software and Systems*, China, May 25-27 2009.

[125] R. G. S. Irani and S. Shukla. Competitive analysis of dynamic power management strategies for systems with multiple power savings states. In *Proceedings of the conference on Design, automation and test in Europe*, pages 117–, 2002.

[126] D. Sacchetto, M. Zervas, Y. Temiz, G. De Micheli, and Y. Leblebici. Resistive programmable through-silicon vias for reconfigurable 3-d fabrics. *Nanotechnology, IEEE Transactions on*, 11(1) :8–11, Jan 2012.

[127] E. Senn, N. Julien, N. Abdelli, D. Elleouet, and Y. Savary. Building and using system, algorithmic, and architectural power and energy models in the fpga design-flow. In *Intl. Conf. on Reconfigurable Communication-centric SoCs 2006*, Montpellier, France, July 2006.

[128] D. Shin and J. Kim. Intra-task voltage scheduling on DVS-enabled hard real-time systems. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 24(10) :1530 – 1549, Sept. 2005.

[129] D. Shin and J. Kim. Fine-grained DVFS using on-chip regulators. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 8, 2011.

[130] S. K. Shukla and R. K. Gupta.i. A model checking approach to evaluating system level dynamic power management policies for embedded systems. In *Proceedings of the Sixth IEEE International High-Level Design Validation and Test Workshop (HLDVT'01), HLDVT '01*, pages 53–, 2001.

[131] H. Sidiropoulos, K. Siozios, and D. Soudris. A novel 3-d FPGA architecture targeting communication intensive applications. *Journal of Systems Architecture - Embedded Systems Design*, 60(1) :32–39, 2014.

[132] K. Siozios, V. F. Pavlidis, and D. Soudris. A novel framework for exploring 3-d fpgas with heterogeneous interconnect fabric. *TRETS*, 5(1) :4, 2012.

[133] The SoCLib project : An open modelling and simulation platform for system on chip design, 2009. http://soclib.lip6.fr/.

[134] L. Sterpone, F. Margaglia, M. Koester, J. Hagemeyer, and M. Porrmann. Analysis of seu effects in partially reconfigurable sopcs. In *Adaptive Hardware and Systems (AHS), 2011 NASA/ESA Conference on*, 2011. 7, 9

[135] V. Tiwari, S. Malik, and A. Wolfe. Power analysis of embedded software : A first step towards software power minimization. In *Transactions on VLSI Systems*, 1994.

[136] N. Tredennick and B. Shimamoto. The inevitability of reconfigurable systems. *Queue*, 1(7) :34–43, Oct. 2003.

[137] D. Tsafrir. The context-switch overhead inflicted by hardware interrupts (and the enigma of do-nothing loops). In *Proceedings of the Workshop on Experimental Computer Science*, 2007.

[138] J. Van Olmen, A. Mercha, G. Katti, C. Huyghebaert, J. Van Aelst, E. Seppala, Z. Chao, S. Armini, J. Vaes, R. Teixeira, M. Van Cauwenberghe, P. Verdonck, K. Verhemeldonck, A. Jourdain, W. Ruythooren, M. De Potter de ten Broeck, A. Opdebeeck, T. Chiarella, B. Parvais, I. Debusschere, T. Y. Hoffmann, B. De Wachter, W. Dehaene, M. Stucchi, M. Rakowski, P. Soussan, R. Cartuyvels, E. Beyne, S. Biesemans, and B. Swinnen. 3d stacked ic demonstration using a through silicon via first approach. In *Electron Devices Meeting, 2008. IEDM 2008. IEEE International*, pages 1–4, Dec 2008.

[139] VectorFabrics : Pareon. Analyze your sequential C code to create an optimized parallel implementation, 2013. http://www.vectorfabrics.com/.

[140] J. R. Villarreal, A. Park, W. A. Najjar, and R. Halstead. Designing modular hardware accelerators in c with roccc 2.0. In *18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines, FCCM 2010, Charlotte, North Carolina, USA, 2-4 May 2010*, pages 127–134, 2010.

[141] J. R. Villarreal, A. Park, W. A. Najjar, and R. Halstead. Designing modular hardware accelerators in c with roccc 2.0. In *18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines, FCCM 2010, Charlotte, North Carolina, USA, 2-4 May 2010*, pages 127–134, 2010.

[142] VITA. Vita57 fmc, 2013.

[143] W. Wolf, A. Jerraya, and G. Martin. Multiprocessor system-on-chip (mpsoc) technology. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 27(10) :1701–1713, Oct 2008. 4

[144] W. Ye, N. Vijaykrishnan, M. Kandemir, and M. Irwin. The Design and Use of SimplePower : A Cycle Accurate Energy Estimation Tool. In *Design Automation Conf*, June 2000.

[145] P. Yuh, C. Yang, C. Li, and C. Lin. Leakage-aware task scheduling for partially dynamically reconfigurable FPGAs. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 14(4) :1–26, 2009.

[146] X. Zhao, Y. Guo, H. Wang, and X. Chen. Fine-grained energy estimation and optimization of embedded operating systems. In *International Conference on Embedded Software and Systems Symposia, ICESS Symposia '08*, pages 90–95, July 2008.

[147] W. H. Zheng, N. Marzwell, and S. Chau. In-system partial run-time reconfiguration for fault recovery applications on spacecrafts. In *Systems, Man and Cybernetics, 2005 IEEE International Conference on*, volume 4, pages 3952–3957 Vol. 4, 2005. 7, 9